commodore

COMPUTER CLUB

8 L. 2.500

La rivista degli utenti di sistemi Commodore

Supplemento a Computer 63 - 30 gennaio 1984 - Sped. Abb. post. gr. III/70 - Distr. Me. Pe spa

Il VIC parla italiano

Giochiamo con 100 racchette

Assembler per tutti

I caratteri speciali dei computer Commodore

Inchiesta tra i lettori

0. Nome (facoltativo): ____________________

1. Indirizzo: via N. città ____________________

2. Come hai avuto questa copia di Commodore Computer Club?

□1 L'ho comprata in edicola. □2 Sono abbonato. □3 Me l'ha passata un amico/familiare. □4 Altro ...

3. Età:

□1 Meno di 15 anni □2 15-18 □3 19-24 □4 25-30

□5 31-45 □6 46-65 □7 oltre 65

4. Sesso:

□1 M □2 F

5. Occupazione:

5.A Gli studenti sono pregati di indicare anche l'occupazione dei genitori.

□1 Padre □2 Madre

6. Se lavorate nel settore dell'informatica completate anche le seguenti caselle, altrimenti GOTO 8:

Indicate l'area precisa in cui operate:

□1 Mainframe □2 Mini □3 Micro

7. **Se Vi occupate in particolare di micro precisate ulteriormente il settore, altrimenti GOTO 8:**

□1 vendita □2 assistenza □3 software □4 altro

8. **Quale delle seguenti pubblicazioni leggete ed apprezzate di più?**

	Leggo Regolarmente	*Qualche volta*	*Mi piace di più*
M&P Computer	□1	□2	□1
Applicando	□1	□2	□2
Basic (enciclopedia)	□1	□2	□3
Bit	□1	□2	□4
Commodore Computer Club	□1	□2	□5
Computer	□1	□2	□6
Elettronica 2000	□1	□2	□7
Enciclopedia dell'informatica	□1	□2	□8
Informatica oggi	□1	□2	□9
Micro Computer	□1	□2	□10
Nuova Elettronica	□1	□2	□11
PC	□1	□2	□12
Personal software	□1	□2	□13
Sinclair Computer	□1	□2	□14
Video Giochi	□1	□2	□15
Zerouno	□1	□2	□16
Altro (precisare)	□1	□2	□17

9. **Quali sezioni della rivista ti interessano di più** ....

11. **Quali argomenti vorreste vedere trattati?**

12. **Quali altri commenti e suggerimenti puoi formulare?**

13. **Possiedi un personal computer?**

□1 SI □2 NO

Se sì, indica la marca ed il modello, altrimenti GOTO 20

MARCA MODELLO da Kbyte

14. **Possiedi un registratore speciale per il tuo computer?**

□1 SI □2 NO

Se no, intendi acquistarne uno nei prossimi 12 mesi?

□3 SI □4 NO

15. **Possiedi un joystick?**

□1 SI □2 NO

Se no, intendi acquistarlo entro i prossimi 12 mesi?

□3 SI □4 NO

16. **Possiedi una manopola (paddle)?**

□1 SI □2 NO

Se no, intendi acquistarla entro i prossimi 12 mesi?

□3 SI □4NNO

17. **Possiedi una tavoletta grafica?**

□1 SI □2 NO

Se no, intendi acquistarla entro i prossimi 12 mesi?

□3 SI □4 NO

18. **Possiedi una penna ottica?**

□1 SI □2 NO

Se no, intendi acquistarla entro i prossimi 12 mesi?

□3 SI □4 NO

19. **Possiedi un'unità a dischetto?**

□1 SI □2 NO

Possiedi una stampante?

□3 SI □4 NO

20. **Intendi comprare un personal computer nei prossimi 12 mesi?**

□1 SI □2 NO

Puoi indicare la marca ed il modello verso cui sei orientato?

21. **Intendi comprare entro l'anno un'unità a dischetti?**

□1 SI □2 NO

22. **Pensi di comprare una stampante entro la fine dell'anno?**

□1 SI □2 NO

23. **Quanto conti di spendere complessivamente?**

□1 Meno di L. 200 mila □2 L. 200 - 500 mila

□3 L. 500 - 999 mila □4 L. 1 - 2 milioni

□5 L. 2 - 5 milioni □6 oltre 5 milioni

24. **Quanti programmi software possiedi?**

Li hai comprati oppure te ne hanno regalato una copia?

□1 Comprati □2 Copiati □3 Regalati

25. **Pensi di comprare dell'altro software?**

□1 SI □2 NO

26. **Che tipo di software comprare?**

☐[1] Giochi ☐[2] Package professionali

☐[3] Software didattico ☐[4] Programmazione di gestione aziendale

27. **Quale sistema operativo usi?**

☐[1] CP/M ☐[2] MS-DOS

☐[3] Altro (specificare)

28. **Compri spesso dei libri sui computer?**

☐[1] Spesso ☐[2] A volte

☐[3] Mai

29. **A cosa ti serve il tuo computer?**

☐[1] In casa / per giocare ☐[2] Per imparare a programmare

☐[3] Per la mia attività / Calcoli scientifici / Progettazione

☐[4] Quale strumento didattico ☐[6] Per lo sviluppo dei sistemi

☐[6] Altro (indicare)

30. **Quante ore alla settimana dedichi al tuo computer?**

☐[1] Quasi mai ☐[2] Meno di 1 ora ☐[3] 1 - 5 ore

☐[4] 6 - 9 ore ☐[5] 10 - 19 ore ☐[6] 20 ore ed oltre

31. **Oltre te, quanti altri familiari o amici usano il tuo computer?**

32. **Oltre te, quanti altri familiari o amici leggono la tua copia di Commodore Computer Club?**

33. **A te capita leggere altre riviste di computer comprate da altri tuoi familiari o amici?**

☐[1] SI ☐[2] NO

Puoi indicare i titoli principali?

Affrancare e spedire a

commodore COMPUTER CLUB

Viale Famagosta 75
20142 Milano

SOMMARIO

Commodore Computer Club - mensile indipendente per gli utenti di sistemi Commodore
Direttore responsabile: Michele di Pisa
Redazione: Alessandro De Simone
Direzione, redazione: V.le Famagosta, 75 - 20142 Milano - Tel. 02 / 8467348
Pubblicità: Milano - Micro Croce, Paola Bevilacqua, Gianluigi Centurelli, Tina Ronchetti, Villa Claudio - V.le Famagosta, 75 - 20142 Milano - Tel. 02 / 8467348/9/40
Prezzi e abbonamenti: Prezzo per una copia Lire 2500
Arretrati il doppio. Abbonamento per dieci fascicoli lire 18.000
Abbonamento annuo cumulativo alle riviste Computer e Commodore Computer Club (tariffa riservata agli studenti) L. 34.000 . I versamenti vanno indirizzati a Minisystem - Italia s.r.l., mediante assegno bancario, vaglia o utilizzando il c/c postale n. 11909207.
Composizioni: Minisystem - Italia
Selezioni: Org. A. G.
Stampa: La Litografica s.r.l. - Busto A.
Registrazione: Tribunale di Milano n. 2/10/1982 - Sped. in abb. post. gr. III n. 70 quale supplemento alla rivista Computer - Pubb. inferiore al 70%

I caratteri speciali dei computer Commodore

SEMPRE più spesso riceviamo richieste di delucidazione sui caratteri "strani" che appaiono sui listati pubblicati. Chiariamo in questa sede, una volta per tutte, il loro significato ed il modo di ottenerli. Teniamo a precisare che il discorso che segue vale per il Vic 20 ed il Commodore 64. Infatti, i modelli della serie 4000 ed 8000 non hanno capacità cromatiche al di fuori del bianco e nero. Cionondimeno i ragionamenti descritti conservano la propria validità anche per questi modelli, pur se limitati ai tasti di controllo cursore, controllo schermo, reverse ed inserimento nuovo carattere.

Passo N. 1

Accendiamo il computer. Dopo alcuni istanti compare un breve messaggio, la scritta READY ed il cursore che lampeggia al di sotto di essa. Se proviamo ora a premere il tasto ← CRSR → (= sinistra-destra, cioè l'ultimo tasto marrone in basso a destra), noteremo che effettivamente il cursore si sposta a destra senza far apparire sullo schermo alcun carattere. Una cosa analoga accade se premiamo il tasto CRSR ↗ up-down↘ (sopra e sotto): a seconda del tasto che premiamo, ed a seconda se premiamo contemporaneamente il tasto SHIFT, riusciamo a spostare il cursore in un punto qualunque dello schermo. E fin qui nulla di nuovo, nemmeno per i principianti.

Passo N.2

Posizionamoci, seguendo i suggerimenti del Passo 1, più o meno al centro dello schermo e premiamo contemporaneamente il tasto SHIFT ed il tasto 2: si visualizzano, in tal modo, le virgolette ("). A questo punto proviamo a premere uno dei due tasti CRSR, insieme, oppure no, al tasto SHIFT; sorpresa! Il cursore non si sposta più nelle quattro direzioni, come visto al punto 1, ma visualizza, a seconda di ciò che digitiamo, uno dei quattro caratteri "strani". Proviamo ora a battere nuovamente le virgolette *una sola altra volta:* subito dopo il cursore riprende ad essere controllato nel modo consueto. Che cos'è successo?

Per spiegarcelo, consideriamo il semplice comando PRINT "PROVA".

Quando il computer interpreta PRINT, "capisce" che deve stampare qualcosa, ed esattamente ciò che è presente dopo la parola-comando PRINT. Quindi, si dispone a decodificare quel "qualcosa". Poichè il primo carattere che incontra è rappresentato dalle virgolette, ne deduce che, deve stampare una stringa o meglio tutti i caratteri che incontrerà tra le virgolette, appena incontrate e le altre che... si augura di incontrare più in là. Poichè, con i computer Commodore è possibile simulare uno spostamento del cursore con la stampa di un carattere speciale, i progettisti del computer hanno pensato: "Se l'utilizzatore della macchina preme il tasto delle virgolette, deve averlo fatto soltanto per definire una stringa, come, ad esempio, PRINT "PROVA", oppure A$="PROVA". Si deduce quindi che, sempre con le virgolette "aperte", se l'utente preme il tasto di controllo cursore (o altri tasti speciali come quelli del colore) non lo fa per spostare fisicamente il cursore sullo schermo, ma solo per comunicare che la stringa che sta per digitare contiene caratteri speciali. A questo punto è dunque necessario che il computer trasformi automaticamente la battitura del tasto di controllo in carattere speciale".

Ma se è vero ciò che è stato appena detto, è anche vero che bisogna tener conto di quando l'utente "chiude" le virgolette e cioè di quando digita nuovamente i tasti SHIFT+2. Ecco dunque spiegato il mistero: dopo aver premuto il tasto delle virgolette un numero di volte DISPARI (=apertura di virgolette), ogni pressione di tasto speciale verrà interpretato dal computer come comando da inserire in una stringa; dopo la pressione di un numero di volte PARI (=chiusura di virgolette), il computer interpreta la pressione di un tasto di controllo come il desiderio, da parte dell'utente, di spostarsi immediatamente da un punto all'altro dello schermo. Si precisa che premendo in qualsiasi momento il tasto Return, con o senza SHIFT, il "contatore" delle virgolette viene azzerato. Inoltre il

computer, nel conteggiare il numero delle virgolette digitate, non tiene conto del fatto che abbiamo, o meno, cancellato alcune di esse con la barra spaziatrice o col tasto INST/DEL. Infine, per i soli modelli della serie 8000 e 600/700, il tasto ESC, se premuto dopo aver digitato le virgolette, annulla la possibilità di far apparire i caratteri-codice alla pressione di tasti particolari.

Quanto detto vale ovviamente per tutti i tasti speciali compresi quelli di cancellazione schermo, definizione colore, reverse eccetera.

Passo N. 3

Proviamo ora ad azzerare il contatore delle virgolette con il tasto di ritorno carrello, e cancelliamo lo schermo (SHIFT + tasto Clr/Home). Premiamo il tasto SHIFT insieme al tasto Inst/Del un paio di volte; apparentemente non succede nulla, tanto è vero che, se premiamo dei tasti alfanumerici questi vengono scritti nel modo consueto come se nulla fosse successo. Proviamo nuovamente a premere SHIFT + Inst/Del un paio di volte e, subito dopo: un tasto di controllo; ci accorgiamo che è come se avessimo premuto le virgolette; vengono trascritti i "codici" dei tasti premuti. Questo è, dunque, un altro modo di visualizzare i caratteri codice tipici degli apparecchi Commodore ed il perchè è presto detto. Se, dopo aver creato uno spazio con SHIFT + Inst/Del, l'utente preme un tasto speciale, non lo fa per utilizzarlo immediatamente, ma di certo per inserirlo, come carattere-codice, all'interno di una stringa precedentemente mal digitata. Si noti che, dopo aver digitato SHIFT + Inst/Del, è possibile visualizzare perfino il carattere-codice della cancellazione carattere (tasto Inst/Del SENZA Shift) che, altrimenti è impossibile da visualizzare.

Concludiamo le presenti note invitando i lettori a studiare la tabella pubblicata che contiene tutti i caratteri speciali della Commodore, specificando i modelli su cui sono presenti. L'ultima stringa dell'elenco contiene tutti i caratteri della lista trascritti l'uno dopo l'altro; consigliamo, comunque i lettori di digitare, come descritto, alla rinfusa i caratteri speciali in modo da abituarsi ad individuarli anche nei listati più... ingarbugliati. ▪

```
***   CARATTERI SPECIALI COMMODORE.  ***
*** SIMBOLOGIA E MODI DI OTTENERLI  ***
***                                  ***
        NERO    CTRL + 1
        BIANCO  '   ' + 2
        ROSSO   '   ' + 3
        AZZURRO'    ' + 4
        VIOLA   '   ' + 5
        VERDE   '   ' + 6
        BLU     '   ' + 7
        GIALLO  '   ' + 8
        REVERSE'    ' + 9
        REV.OFF'    ' + 0
        HOME <CLR/HOME>
        CRSR IN BASSO
        CRSR A DESTRA
        SHIFT + CLR/HOME
          '  ' + CRSR IN ALTO
          '  ' + CRSR A SINISTRA
          '  ' + INST/DEL
        F1
        F3
        F5
        F7
        SHIFT + F1  (F2)
          '  ' + F3  (F4)
          '  ' + F5  (F6)
          '  ' + F7  (F8)
 COLORI. SOLO PER COMMODORE 64
        ARANCIO TASTO COMMODORE + 1
        MARRONE  ' '       '   ' + 2
        ROSSO SC.' '       '   ' + 3
        GRIGIO 1 ' '       '   ' + 4
        GRIGIO 2 ' '       '   ' + 5
        VERDE SC.' '       '   ' + 6
        BLU SCURO' '       '   ' + 7
        GRIGIO 3 ' '       '   ' + 8
```

100 Racchette

Un vecchio gioco molto divertente adattato ai due più noti personal Commodore.

IL GIOCO che presentiamo, era famoso qualche anno fa e, faceva impazzire i possessori dei Pet. Richiede, infatti, una certa prontezza di riflessi perchè, se si sbaglia a premere il tasto giusto al momento giusto, la difficoltà del gioco aumenta.

In che cosa consiste il gioco: impartito il consueto RUN appaiono alcune domande relative al numero di giocatori (fino a 9), al loro nome e alla difficoltà per ciascuno di essi. Esaurita questa prima fase preliminare, lo schermo si cancella e dopo un po' di tempo compare una cornice, che rappresenta i limiti del campo da gioco, una piccola "o" che rappresenta il bersaglio ed una palla che si muove verticalmente rimbalzando ed invertendo il movimento tutte le volte che urta contro i bordi del campo.

Il gioco consiste nel colpire il bersaglio mobile con la palla. Questa può essere controllata con i tasti "M" ed "N" che, come il lettore può verificare, visualizzano due barre inclinate (vedi fig. 1).

Non appena si preme uno dei due tasti, compare sullo schermo una racchetta con inclinazione corrispondente a ciò che appunto risulta stampigliato sul tasto stesso. Attenzione, però, l'effetto della "racchetta" è completamente diverso a seconda della direzione di provenienza della palla (vedi fig. 2). Non solo: la racchetta che viene visualizzata rimane presente sullo schermo aumentando i rimbalzi della palla e, di conseguenza, la difficoltà di centrare il bersaglio. ▪

Figura 1

Figura 2

COMMOD

Se stai comprando un personal computer prova a farti queste domande:
1. Chi è oggi il più affidabile?
2. Chi dà la possibilità di scegliere fra più sistemi?
3. Chi fornisce soluzioni, subito, in una gamma vastissima?
4. Chi propone il miglior rapporto fra costi e prestazioni?
5. Chi ti dà una così grande esperienza ed assistenza?
A tutte le domande puoi rispondere con

ORE, IL N°1

un solo nome: Commodore Computer.
Anche per questo Commodore
è il Numero 1. In Europa e in Italia.
Sei in buone mani.
Commodore Italiana Spa
Milano, telefono 02/6125651

commodore
COMPUTER

ETHOS

```
180 PX=1:DIMN%(9),PN$(9),PS%(9),PM%(9):KK$="      "
190 PRINT"  100 RACCHETTE
200 PRINT"COLPISCI LA PALLA (O)"
210 PRINT"CON LE RACCHETTE N E M"
220 PRINT"N GIOCATORI (1-9)? ";:GOSUB1030:PY=X
230 PRINT"RETURN DOPO OGNI NOME":GOSUB1090
240 PRINT:PRINT:PRINT"PREMI UN TASTO"
250 GETA$:IFA$=""THEN250
260 PL=PL+1:IFPL>PYTHENPL=1
270 B$="                    ":V$="                    "
280 TS=TI:TX=TS+6000:PC=1:PK=4200:D=1
290 PRINT"":FORI=37888TO38399:POKEI,0:NEXT
300 PRINTB$"GIOC."PL" - "LEFT$(PN$(PL),10)"   ";
310 FORI=1TO21:PRINT"                      ";:NEXT
320 PRINT"K RESA=!)   DIFF.="(20-N%(PL))/2""LEFT$(B$,4);
330 T=INT(RND(1)*511+4096):IF(4196 <T)AND(T<4507)THEN330
340 TD=1:NN=N%(PL)
350 IFPEEK(T)<>32THEN330
360 POKET,87
370 PK=PK+D:D=SGN(D)*((ABS(D)AND1)*22+(ABS(D)AND22)/22)
380 FORII=1TONN:IFPEEK(PE)=81THENPOKEPE,32
390 PE=PK:GETI$:IFI$<>""THEN490
400 IFPEEK(PK+D)<>32THEN440
410 PK=PK+D:POKEPK,81:NEXT:POKET,32:T=T+TD*D
420 IFPEEK(T)<>32THENTD=-TD:T=T+TD*D
430 POKET,87:GOTO380
440 IFTI>TXTHENPS=0:GOTO630
450 IFPEEK(PK+D)=78THENPK=PK+D:D=D-SGN(D)*23:GOSUB1250:GOTO380
460 IFPEEK(PK+D)=77THENGOSUB1250:GOTO370
470 D=-D:IFPEEK(PK-D)=87THEN620
480 GOTO380
490 IFI$="N"THEN530
500 IFI$="M"THEN560
510 IFI$="!"THENPS=0:GOTO630
520 GOTO400
530 O=D:PC=PC+1:POKEPK,78
540 D=D-SGN(D)*23:GOTO380
550 IFPEEK(PK+D)=78THENPK=
    PK+D
560 O=D:PC=PC+1:POKEPK,77
570 IFO=-1THEND=-22
580 IFO=22THEND=1
590 IFO=1THEND=22
600 IFO=-22THEND=-1
610 GOTO380
620 PS=1000-12*(PC-1)-INT((TI-TS)/6):IFPS<0THENPS=1
630 PS%(PL)=PS
640 PM%(PL)=INT((PM%(PL)*(PX-1)+PS)/PX)
650 PRINT"┌──────┬──┬──────┬──────┐";
660 PRINT"|GIOC.|DF|PUNT.|MEDIA|";
670 PRINT"├──────┼──┼──────┼──────┤";
680 PRINT"|     |  |     |   |";
```

```
690 FORI=1TOPY
695 REM REGOLO "_-_-_-_-_/_-_-_-_-_/_-_-_-_-_/_-
700 PRINT"|"PN$(I);RIGHT$(KK$+STR$((20-N%(I))/2),2);
710 PRINT"|"RIGHT$(KK$+STR$(PS%(I)),3)"  |  ";
720 PRINT RIGHT$(KK$+STR$(PM%(I)),3)" |";
730 NEXT
740 PRINT"|     |  |     |   |";
750 PRINT"└──────┴──┴──────┴──────┘"
760 PRINT"NUMERO DI PARTITA ="PX:PRINT"";
```

```
770 FORKK=1TO4+PL:PRINT"█";:NEXT:PRINT"*█":FORKK=PLTOPY:PRINT"█";:NEXT
780 IFPL=PYTHENGOSUB800:FORI=1TOPY:PS%(I)=0:NEXT:PX=PX+1:GOTO240
790 PRINT"█████PREMI UN TASTO PER CONTINUARE":GOTO250
800 PRINT"█████CAMBI ██D█IF██F█ICOLTA'(S/N)?█ ";:GOSUB970
810 IFA$="N"THENRETURN
820 PRINT"████GIOCATORE NUMERO ?";:GOSUB1030:XX=X:IFX>PYTHEN820
830 A$=PN$(XX):AA=10
840 IFMID$(A$,AA,1)=" "ANDAA>1THENAA=AA-1:GOTO840
850 A$=LEFT$(A$,AA):PRINT"███"A$:PRINT" DIFFICOLTA' ATTUALE :"(20-N%(XX))
*   /2:PRINT
860 PRINT" LA MEDIA E'"PM%(XX):PRINT
870 PRINT" PUNT. PREC."PS%(XX)
880 PRINT"█NUOVA █D█I█F█FICOLTA'(1-9)? ";:GOSUB1030:N%(XX)=20-X*2
890 PRINT:PRINT"█CAMBI ANCORA(S/N)? ";:GOSUB970:IFA$="S"THEN820
900 RETURN
910 FORI=1TO100:GETA$:IFA$<>""THEN950
920 NEXT:PRINT"█ █";
930 FORI=1TO100:GETA$:IFA$<>""THEN950
940 NEXT:PRINT"█ █";:GOTO910
950 IFVAL(A$)=0THEN910
960 PRINT"█"A$:S=VAL(A$):RETURN
970 FORI=1TO100:GETA$:IFA$<>""THEN1010
980 NEXT:PRINT"█ █";
990 FORI=1TO100:GETA$:IFA$<>""THEN1010
1000 NEXT:PRINT"█ █";:GOTO970
1010 IFA$<>"S"ANDA$<>"N"THEN970
1020 PRINT"█"A$:RETURN
1030 FORI=1TO100:GETA$:IFA$<>""THEN1070
1040 NEXT:PRINT"█ █";
1050 FORI=1TO100:GETA$:IFA$<>""THEN1070
1060 NEXT:PRINT"█ █";:GOTO1030
1070 IFA$<"1"ORA$>"9"THEN1030
1080 PRINT"█"A$:X=VAL(A$):RETURN
1090 PRINT"█":FORK=1TOPY:PRINT:PRINT
1100 PRINT"NOME DEL GIOCATORE"K"? ____________██████████";
1110 GOSUB1140:PRINT
1120 PRINT:PRINT"DIFFICOLTA' PER IL GIOCAT."K"(1-9)? ";:GOSUB1030:N%(K)
1130 NEXT:RETURN                                                   =20-2*X
1140 B$=""
1150 FORJ=1TO50:GETA$:IFA$<>""THEN1180
1160 NEXT:PRINT"█_█";:FORJ=1TO50:GETA$:IFA$<>""THEN1180
1170 NEXT:PRINT"█_█";:GOTO1150
1180 A=ASC(A$):IFA=13THENPRINT"█_":GOTO1230
1190 IFA=20ANDLEN(B$)>1THENB$=LEFT$(B$,LEN(B$)-1):PRINT"█_██_█";:GOTO1150
1200 IFA=20ANDLEN(B$)=1THENB$="":PRINT"█_██_█";:GOTO1150
1210 IFA=32OR(64<AANDA<98)THENPRINT"█"A$;:B$=B$+A$:IFLEN(B$)>9THEN1230
1220 GOTO1150
1230 B$=B$+"           ":PN$(K)=LEFT$(B$,5)+"|":RETURN
1240 REM *** SEZIONE SONORA ***
1250 POKE36878,15:POKE36876,200:FORXX=1TO10:NEXT
1260 POKE36878,0:RETURN
1270 REM REGOLO "_-_-_-_-_-/_-_-_-_-_-/_-_-_-_-_-/_-_-_-_-_-"
```

```
165 REM ****VER. COMMODORE 64 ***
167 REM ****                    ***
170 REM ************************
180 :
190 PX=1:DIMN%(9),PN$(9),PS%(9),PM%(9)
200 PRINT"              100 RACCHETTE"
210 PRINT"CERCA DI COLPIRE IL BERSAGLIO (= O)"
220 PRINT"PREMENDO LE 'RACCHETTE'  N  E  M "
230 PRINT"NUMERO DEI GIOCATORI (1-9)? ";:GOSUB1020:PY=X
240 PRINT"PREMI RETURN DOPO OGNI NOME":GOSUB1080
250 PRINT:PRINT:PRINTTAB(7)"PREMI UN TASTO PER INIZIARE"
260 GETA$:IFA$=""THEN260
270 PL=PL+1:IFPL>PYTHENPL=1
280 B$="                                        ":V$="                              "
290 TS=TI:TX=TS+6000:PC=1:PK=1100:D=1
300 PRINT"":FORI=55296TO56295:POKEI,1:NEXT
310 PRINT""B$"GIOCATORE "PL" - "LEFT$(PN$(PL),10)SPC(12);
320 FORI=1TO22:PRINT"                                      ";:NEXT
330 PRINT"PREMI! PER ABBANDONARE.DIFFICOLTA'="(20-N%(PL))/2""LEFT$(B$,4);
340 T=INT(RND(1)*900+1024):IF(33248<T)AND(T<33288)THEN340
350 TD=1:NN=N%(PL)
360 IFPEEK(T)<>32THEN340
370 POKET,87
380 PK=PK+D:D=SGN(D)*((ABS(D)AND1)*40+(ABS(D)AND40)/40)
390 FORII=1TONN:IFPEEK(PE)=81THENPOKEPE,32
400 PE=PK:GETI$:IFI$<>""THEN500
410 IFPEEK(PK+D)<>32THEN450
420 PK=PK+D:POKEPK,81:NEXT:POKET,32:T=T+TD*D
430 IFPEEK(T)<>32THENTD=-TD:T=T+TD*D
440 POKET,87:GOTO390
450 IFTI>TXTHENPS=0:GOTO640
460 IFPEEK(PK+D)=78THENPK=PK+D:D=D-SGN(D)
    *41:GOSUB1240:GOTO390
470 IFPEEK(PK+D)=77THENGOSUB1240:GOTO380
480 D=-D:IFPEEK(PK-D)=87THEN630
490 GOTO390
500 IFI$="N"THEN540
510 IFI$="M"THEN570
520 IFI$="!"THENPS=0:GOTO640
530 GOTO410
540 O=D:PC=PC+1:POKEPK,78
550 D=D-SGN(D)*41:GOTO390
560 IFPEEK(PK+D)=78THENPK=PK+D
570 O=D:PC=PC+1:POKEPK,77
580 IFO=-1THEND=-40
590 IFO=40THEND=1
600 IFO=1THEND=40
610 IFO=-40THEND=-1
620 GOTO390
630 PS=1000-12*(PC-1)-INT((TI-TS)/6):IFPS<0THENPS=1
640 PS%(PL)=PS
650 PM%(PL)=INT((PM%(PL)*(PX-1)+PS)/PX)
660 PRINT"┌──────────┬─────┬──────┬───────┐"
670 PRINT"|  GIOCAT.  |DIFF.| PUNT. | MEDIA |"
680 PRINT"├──────────┼─────┼──────┼───────┤"
690 PRINT"|           |     |      |       |"
700 FORI=1TOPY
705 REM REGOLO "_-_-_-_-_-/_-_-_-_-_-/_-_-_-_-_-/_-_-_-_-_-/
710 PRINT"| "PN$(I)(20-N%(I))/2" | "PS%(I)TAB(29)"| "PM%(I)TAB(37)"|"
720 NEXT
730 PRINT"|           |     |      |       |"
740 PRINT"└──────────┴─────┴──────┴───────┘"
750 PRINTSPC(10)"NUMERO DI PARTITA ="PX:PRINT"";
```

```
760 FORKK=1TO4+PL:PRINT"■";:NEXT:PRINT"■■■■*":FORKK=PLTOPY:PRINT"■";:NEXT
770 IFPL=PYTHENGOSUB790:FORI=1TOPY:PS%(I)=0:NEXT:PX=PX+1:GOTO250
780 PRINT"■■■■"SPC(8)"■PREMI UN TASTO PER CONTINUARE":GOTO260
790 PRINT"■■■■"SPC(7)"■CAMBI DIFFICOLTA'   (S/N)?■ ";:GOSUB960
800 IFA$="N"THENRETURN
810 PRINT"■■■■GIOCATORE NUMERO ?";:GOSUB1020:XX=X:IFX>PYTHEN810
820 A$=PN$(XX):AA=10
830 IFMID$(A$,AA,1)=" "ANDAA>1THENAA=AA-1:GOTO830
840 A$=LEFT$(A$,AA):PRINT"■■■"A$"■ DIFFICOLTA' ATTUALE :"(20-N%(XX))/2:PRINT
850 PRINTSPC(LEN(A$))" LA MEDIA RAGGIUNTA E'"PM%(XX):PRINT
860 PRINTSPC(LEN(A$))" PUNTEGGIO PRECEDENTE "PS%(XX)
870 PRINT"■NUOVA DIFFICOLTA'    (1-9)? ";:GOSUB1020:N%(XX)=20-X*2
880 PRINT:PRINT"■CAMBI ANCORA   (S/N)? ";:GOSUB960:IFA$="S"THEN810
890 RETURN
900 FORI=1TO100:GETA$:IFA$<>""THEN940
910 NEXT:PRINT"■ ■";
920 FORI=1TO100:GETA$:IFA$<>""THEN940
930 NEXT:PRINT"■ ■";:GOTO900
940 IFVAL(A$)=0THEN900
950 PRINT"■"A$:S=VAL(A$):RETURN
960 FORI=1TO100:GETA$:IFA$<>""THEN1000
970 NEXT:PRINT"■ ■";
980 FORI=1TO100:GETA$:IFA$<>""THEN1000
990 NEXT:PRINT"■ ■";:GOTO960
1000 IFA$<>"S"ANDA$<>"N"THEN960
1010 PRINT"■"A$:RETURN
1020 FORI=1TO100:GETA$:IFA$<>""THEN1060
1030 NEXT:PRINT"■ ■";
1040 FORI=1TO100:GETA$:IFA$<>""THEN1060
1050 NEXT:PRINT"■ ■";:GOTO1020
1060 IFA$<"1"ORA$>"9"THEN1020
1070 PRINT"■"A$:X=VAL(A$):RETURN
1080 PRINT"■":FORK=1TOPY:PRINT:PRINT
1090 PRINT"NOME DEL GIOCATORE"K"? ______________ ■■■■■■■■■■■";
1100 GOSUB1130:PRINT
1110 PRINT:PRINT"DIFFICOLTA' PER IL GIOCAT."K"(1-9)? ";:GOSUB1020:N%(K)=20-2*X
1120 NEXT:RETURN
1130 B$=""
1140 FORJ=1TO50:GETA$:IFA$<>""THEN1170
1150 NEXT:PRINT"■_■";:FORJ=1TO50:GETA$:IFA$<>""THEN1170
1160 NEXT:PRINT"■_■";:GOTO1140
1170 A=ASC(A$):IFA=13THENPRINT"■_":GOTO1220
1180 IFA=20ANDLEN(B$)>1THENB$=LEFT$(B$,LEN(B$)-1):PRINT"■_■■_■";:GOTO1140
1190 IFA=20ANDLEN(B$)=1THENB$="":PRINT"■_■■_■";:GOTO1140
1200 IFA=32OR(64<AANDA<98)THENPRINT"■"A$;:B$=B$+A$:IFLEN(B$)>9THEN1220
1210 GOTO1140
1220 B$=B$+"          ":PN$(K)=LEFT$(B$,10)+"| ":RETURN
1230 REM *** SEZIONE SONORA ***
1240 POKE 54296,15
1250 POKE 54277,64: POKE 54276,17 :POKE54272,17:POKE54273,87
1260 FOR K=1 TO 10: NEXT:POKE 54296,0
1270 RETURN   1280 REM REGOLO "_‾_"
```

Il buffer della tastiera

Cerchiamo di capire come "ragiona" il nostro personal.

L'ARTICOLO illustra, in modo più che elementare, ciò che accade tutte le volte che premiamo un tasto. Come applicazione pratica si esaminerà un programma che, a dispetto della sua brevità, può essere utilizzato in più di un'occasione.

Se il lettore avrà la pazienza di seguire alla lettera tutte le note che seguono, sarà in grado di apprendere nuove nozioni in modo semplice e rapido.

• *Prima fase.* Digitate il seguente microprogramma:

100 TI$ = "000000" FOR I = 1 TO 7000 : NEXT: PRINT TI$

Esso non fa altro che azzerare l'orologio interno del computer, contare da 1 a 7000 e visualizzare il tempo impiegato nel conteggio. Questo risulta essere di circa sei secondi.

• *Seconda fase.* Facciamo partire il programma (RUN) e con una certa rapidità, prima cioè che trascorrano i sei secondi, digitiamo sette tasti a caso, come ad esempio: QWERTYU

Possiamo notare che essi non vengono visualizzati immediatamente sullo schermo. Quando però, il tempo dell'elaborazione termina, essi appaiono sul video e il cursore lampeggia a destra del settimo carattere.

Ciò dimostra che i tasti premuti, durante un'elaborazione, non vengono "persi", ma memorizzati da qualche parte all'interno della memoria del computer e stampati non appena l'elaborazione termina.

• *Terza fase.* Battiamo nuovamente RUN e, prima che il conteggio abbia termine, digitiamo tutta la seconda fila della tastiera (13 tasti):

QWERTYUIOP@*↑

Quando però il programma termina noteremo che vengono visualizzati solo i primi dieci caratteri (QWERTYUIOP). Ciò dimostra che non è possibile memorizzare più di dieci caratteri contemporaneamente.

Il luogo in cui i caratteri vengono memorizzati è detto reyboard buffer: memoria di transito della tastiera.

• *Quarta fase.* Il buffer di tastiera è localizzato nei dieci byte il cui indirizzo va da 631 a 640. Il numero di caratteri da considerare è memorizzato nel byte 198.

Se per esempio, in un certo istante, in 198 è presente il valore 5, questo comunica al sistema operativo del computer che, i caratteri del buffer da considerare sono i primi cinque e che gli altri, eventualmente presenti, devono essere ignorati.

• *Quinta fase.* Digitate ora il programma ESAME BUFFER TASTIERA. Commentiamolo brevemente. Il contenuto della locazione 653 vale uno nel caso sia premuto il tasto SHIFT, due se si preme il tasto Commodore, quattro se si preme CTRL. Ovviamente, vale zero nel caso non venga premuto nessuno dei tre tasti. Nella locazione 197, invece, è contenuto il valore-codice degli altri tasti della tastiera. In questo caso, però, se non viene premuto alcun tasto, nella locazione 197

```
90 REM *** ESAME BUFFER TASTIERA ***
92 :
95 IF PEEK(653)=4 THEN GOSUB 150: REM AZZERAMENTO: TASTO CTRL
100 IF PEEK(653)=2 THEN GOSUB 130: REM VISUALIZZAZIONE: TASTO COMM.
110 IF PEEK(197)=39 THEN END:REM FINE: TASTO F1 PER VIC 20
115 IF PEEK(197)=4 THEN END:REM FINE: TASTO F1 PER COMMODORE 64
120 GOTO 95
125 :
130 FOR I=0TO 9: PRINT PEEK(631+I);: NEXT
140 PRINT "■" PEEK(198): RETURN
145 :
150 POKE 198,0
160 FOR I=0TO 9: POKE 631+I,0: NEXT: RETURN
```

sarà contenuto il valore 64. Il valore 39 corrisponde al tasto f1. Si fa notare esplicitamente che i valori non corrispondono al codice ASCII, ma ad un codice particolare Commodore. Il lettore che desidera conoscerlo può digitare a parte il seguente... programma:

10 PRINT PEEK(197):
GOTO 10

Facciamo partire il programma ESAME BUFFER TASTIERA: premendo il tasto Commodore, verrà visualizzato il contenuto dei dieci byte e, in reverse, quello della locazione 198. Premendo CTRL, invece, il buffer viene interamente azzerato. Con f1, infine, il programma termina.

Premiamo alcuni tasti a caso ed in seguito il tasto Commodore, oppure il CTRL. Possiamo notare i vari cambiamenti che si susseguono. Ciò dimostra che (vedi subroutine 150) possiamo modificare a piacimento il contenuto del buffer.

• *Sesta fase.* Il programma INTERP A$ non è altro che una pratica applicazione di quanto esaminato finora.

Alcuni linguaggi Basic possiedono un'istruzione molto potente: INTERP A$. Vale a dire che:

A$ = "PRINT 6*6+4"

il comando INTERP A$ fornisce come risultato: 40.

I modelli Commodore non possiedono tale istruzione ed il programma proposto simula il comando.

Funzionamento del programma

Quando (riga 100) viene visualizzata la domanda: ORDINE? rispondete con una frase s intatticamente corretta, come ad esempio:

PRINT 4*5+89

Subito dopo (riga 110) lo schermo viene cancellato, e la stringa A$ visualizzata in alto a sinistra. Inoltre, a qualche rigo di distanza, viene stampato due volte il comando CONT. La riga 120 comunica al Sistema Operativo che dovrà considerare, non appena il programma termina (riga 135) i primi quattro byte presenti nel buffer.

La riga 130 "scrive" nel buffer quattro valori che corrispondono a HOME (19) e RETURN (13). L'ultimo byte deve essere sempre uno zero. Fatto questo il programma termina (END riga 135) ed il S.O. provvede ad eseguire i comandi presenti nel buffer che, guarda caso, per come è stato precedentemente strutturato, simula per due volte la pressione del tasto RETURN: la prima per eseguire l'ordine rappresentato da A$, e la seconda per impartire il comando CONT, precedentemente stampato sullo schermo. Con CONT il programma prosegue alla linea successiva (140) che rinvia alla 100, ripetendo il ciclo.

Avvertenza

L'istruzione INPUT non accetta, come carattere, la virgola (,) il punto e virgola (;) e i due punti (:). Desiderando far interpretare comandi che li contengono è necessario, alla domanda ORDINE, digitare come primo carattere, le virgolette ("). Esempio:

ORDINE? "A=13: B=2: PRINT A+B ▪

Alessandro De Simone

```
10 REM ***  SIMULAZIONE DELLA  ***
20 REM *** FUNZIONE: INTERP A$ ***
30 :
90 PRINT "": B$="                     ": REM 21 SPAZI
100 PRINT B$ "" B$: INPUT "ORDINE";A$
105 IF A$="←" THEN END: REM A.DE SIMONE
110 PRINT "" A$: PRINT: PRINT: PRINT "CONT": PRINT "CONT"
120 POKE 198,4
130 POKE 631,19: POKE 632,13: POKE 633,13: POKE634,0
135 PRINT:PRINT:PRINT:END
140 GOTO100

READY.
```

** BIS **

QUESTO programma riproduce l'ormai famoso gioco condotto da Mike Buongiorno su una nota emittente privata italiana. Per chi non conoscesse il gioco, diamone un breve cenno. *I due concorrenti possono scoprire a turno due caselle del tabellone sotto ad ogni casella comparirà un simbolo e si deve cercare di fare il maggior numero di coppie (più di nove), altrimenti la partita è patta.*

Il programma necessita di almeno 3K RAM aggiuntivi o della S.E. (non necessita di modifiche se usato con più di 8K).

Dato il (RUN) vengono chiesti i nomi dei due concorrenti (per ragioni di schermo deve essere al massimo di dieci caratteri), viene poi disegnato il tabellone con ogni casella, rappresentata da un numero da 00 a 35. (Attenzione che i numeri da 0 a 9 sono scritti preceduti da uno zero, esclusivamente per ragioni di presentazione grafica, non sarà dunque, necessario digitare questo zero scegliendo la casella).

Al primo dei due concorrenti verrà chiesto di scegliere fra testa e croce (tasti "T" o "C"), viene lanciata la moneta e chi vince gioca per primo. Ora viene chiesto al giocatore in input il numero della prima casella che vuole scoprire (nel caso che la casella sia già stata scoperta l'input non viene accettato e la domanda ripetuta), viene così, scoperta la casella e si attende il numero della seconda con le stesse modalità.

Nel caso che sia stata fatta una coppia, questa viene segnalata con una scritta lampeggiante in basso al teleschermo e le due caselle vengono colorate del colore del concorrente che ha indovinato; logicamente, il gioco rimane in mano a questi fino a quando non sbaglia.

Se non viene fatta la coppia, le due caselle rimangono scoperte per un dato tempo, dopo di che, vengono rigirate ed il gioco passa all'avversario. A differenza del gioco di Mike, le coppie sono effettive e non c'è il jolly. Terminate tutte le 18 coppie del tabellone, viene proclamato il vincitore.

Per chi non si accontenta dei simboli un po' scarni sotto alle caselle con un certo qual lavoro di pazienza è possibile ridefinire i caratteri e disegnare per esempio: aerei, navi, auto ecc.

Un consiglio, affinchè la casella girata sia più facilmente individuabile suggerisco di tenere il tabellone tutto in scuro e ridefinire i caratteri in modo normale.

Enrico Franco

```
90 :
100 DIMA$(36),B%(36),HH$(36)
110 CC$(1)="█▒  ▀▀":CC$(0)="◣▒  ▀▀"
120 B$="                    ":KK=36:TD=2000
130 H$="████████████████████"
140 YY$(1)="██":XX$(1)="██"
150 YY$(2)="█████":XX$(2)="█████"
160 YY$(3)="████████":XX$(3)="████████"
170 YY$(4)="███████████":XX$(4)="███████████":YY$(5)="██████████████":XX$(5)=
"██████████ ]]]]]"
180 YY$(6)="████████████████":XX$(6)="████████████████████"
190 GOSUB550
200 GOSUB1180:GOSUB620
210 GOSUB1210
220 GOSUB750
230 GOSUB960
240 GOSUB750
250 GOTO410
260 PRINTH$;"██CASELLA N°:██";
270 INPUTA:K=A:GOSUB830
280 IFER=1THENPRINTH$;B$:GOTO260
290 K=A:GOSUB800
300 PRINTH$;"██CASELLA N°:██";
310 INPUTB:K=B:GOSUB830
320 IFER=1THENPRINTH$;"██";B$;"□□";:GOTO300 →
```

```
330 K=B:GOSUB800
340 GOSUB750
350 IFA$(A)=A$(B)THEN430
360 FORT=1TO1000:NEXT
370 K=A:GOSUB770
380 K=B:GOSUB770
390 IFA$=AA$(0)THENA$=AA$(1):J=1:GOTO410
400 A$=AA$(0):J=0
410 PRINTH$;"■ TOCCA A :■  ";A$
420 GOTO260
430 B%(A)=1:B%(B)=1:GOSUB910:GOSUB1100
440 BB(J)=BB(J)+1
450 N=N-1:IFN=0THEN470
460 GOTO410
470 IFBB(0)=BB(1)THENGOSUB750:PRINTH$;"■ PARTITA PATTA !■":GOTO510
480 GOSUB750:PRINTH$;"■HA VINTO ■";
490 IFBB(0)>BB(1)THENPRINTAA$(0):GOTO510
500 PRINTAA$(1)
510 PRINT"■■  ALTRA GARA ? (Y/N)"
520 GETA$:IFA$=""THEN520
530 IFA$="Y"ORA$="S"THENPRINT"■":RUN
540 PRINT"■":END
550 PRINT"■NOME PRIMO CONCORRENTE■■"
560 INPUTAA$(0)
570 IFLEN(AA$(0))>10THENPRINT"■TROPPO LUNGO !■":GOTO560
580 PRINT"■NOME ALTRO CONCORRENTE■■"
590 INPUTAA$(1)
600 IFLEN(AA$(1))>10THENPRINT"■TROPPO LUNGO !■":GOTO590
610 PRINT"■":RETURN
620 N=18:PRINT"■ ■                    ■"
630 FORJ=1TO6:PRINT"■";
640 FORI=1TO6
650 PRINT"■ ■  ";
660 NEXT
670 PRINT"■ ■"
680 PRINT" ■                    ■"
690 NEXT
700 PRINT" ■                    ■"
710 FORJ=0TO35
720 K=J:GOSUB770
730 NEXT
740 RETURN
750 PRINTH$;
760 FORHH=1TO6:PRINTB$:NEXT:RETURN
770 Y=INT(K/6)+1:X=K-((Y-1)*6)+1
780 PRINTYY$(Y);XX$(X);HH$(K)
790 B%(K)=0:RETURN
800 Y=INT(K/6)+1:X=K-((Y-1)*6)+1
810 PRINTYY$(Y);XX$(X);A$(K)
820 B%(K)=1:RETURN
830 ER=0
```

```
840  IFK>=36THENER=1:RETURN
850  IFNOT(KK<>36)THEN880
860  IFK=KKTHENER=1:RETURN
870  KK=36:GOTO890
880  KK=K
890  IFB%(K)=1THENER=1:RETURN
900  RETURN
910  K=A:GOSUB940
920  K=B:GOSUB940
930  RETURN
940  Y=INT(K/6)+1:X=K-((Y-1)*6)+1
950  PRINTYY$(Y);XX$(X);CC$(J):RETURN
960  PRINTH$;AA$(0)
970  PRINT"■T■ESTA O ■C■ROCE ?"
980  T=1
990  GETA$:IFA$=""THEN990
1000  IFA$="T"THEN1030
1010  IFA$="C"THEN1030
1020  GOTO990
1030  A=INT(RND(0)*100)
1040  IFA>50THENPRINT"■■TESTA■":T=1:GOTO1060
1050  PRINT"■■CROCE■"
1060  PRINT"■INIZIA ";:T=0
1070  IFT=1AND A$="T"THENPRINTAA$(0):J=0:A$=AA$(0):GOTO1090
1080  PRINTAA$(1):J=1:A$=AA$(1)
1090  FORHH=1TOTD:NEXT:T=0:RETURN
1100  GOSUB750:HH=0
1110  PRINTH$;"■■■COPPIA■"
1120  HH=HH+1
1130  FORT=1TO200:NEXT
1140  PRINTH$;"■■COPPIA"
1150  FORT=1TO200:NEXT
1160  IFHH>=5THENGOSUB750:RETURN
1170  GOTO1110
1180  FORJ=0TO35:READA$:HH$(J)=A$:NEXT:RETURN
1190  DATA00,01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20
1200  DATA21,22,23,24,25,26,27,28,29,30,31,32,33,34,35
1210  PRINTH$;"■ATTENDERE PREGO":FORJ=0TO35:A$(J)="":NEXT
1220  FORJ=0TO17:PRINTH$;"■■";J
1230  READA$
1240  Z=INT(RND(1)*36)
1250  IFA$(Z)<>""THEN1240
1260  A$(Z)="■"+A$+"■"
1270  Z=INT(RND(1)*36)
1280  IFA$(Z)<>""THEN1270
1290  A$(Z)="■"+A$+"■"
1300  NEXT
1310  RETURN
1320  DATAAA,BB,**,EE,FF,##,%%,II,$$,&&,??,◀◀,££
1330  DATA"●●","OO","♠♠","♥♥","♦♦","♣♣"
1340  END
```

Franco Enrico
via Marco Polo, 1
10095 Crugliasco (To)

IMPARA A PROGRAMMARE CON IL VIC

★ ★ ★ ★ ★ ★ ★ ★ ★ ★ ★

Dispensa n. 8

Subroutine e grafica ad alta risoluzione

Finalità delle subroutine

La subroutine è una sequenza di istruzioni studiate per svolgere uno o più compiti specifici, che può essere necessaria più volte e in più parti dello stesso programma. Una routine scritta sotto forma di subroutine, viene incorporata nel programma una sola volta. Durante la sua esecuzione, l'istruzione

GOSUB numero di linea

provoca il passaggio del controllo al numero di linea specificato e l'esecuzione continua fino all'incontro con la prima istruzione RETURN, dopodichè il controllo ritorna all'istruzione che viene immediatamente dopo l'istruzione GOSUB che ha dato origine al primo trasferimento di controllo.

La subroutine può essere introdotta tutte le volte che è necessaria, consentendo così un enorme risparmio di tempo là dove un certo numero di istruzioni si ripetono più volte nel corso dello stesso programma. Di solito, quando non si fa uso di subroutine, il programma tende a diventare molto più lungo e, si sa, un programma più lungo ha bisogno di un'area di memoria più grande e richiede più tempo per essere tradotto in linguaggio macchina. Inoltre, l'uso delle subroutine rende il programma più efficace.

Una subroutine può essere ricopiata ed usata in programmi molto diversi fra loro, sia nella sua forma originale che in forma modificata. E' opportuno, perciò, che le subroutine siano sviluppate in modo da poter essere usate in più modi possibili senza modifiche: occorre cioè che siano molto flessibili.

Sviluppo indipendente

Un altro vantaggio connesso con l'uso delle subroutine è la possibilità che esse offrono di poter essere sviluppate e provate in modo autonomo rispetto al programma nel corso del quale dovranno essere utilizzate.

Il collaudo autonomo delle subroutine permette di realizzare programmi molto complessi più rapidamente, attraverso un'operazione di "montaggio". Inoltre, per sua natura, la subroutine sviluppata per un programma può essere collaudata per l'uso in altri programmi con opportuni dati di prova prima di essere inserita nel programma cui è destinata. Il programma finale ha comunque bisogno di essere ulteriormente collaudato come insieme omogeneo per verificare che le parti del collegamento, cioè le istruzioni tra le diverse subroutine (così come tutte le subroutine inserite), forniscano risultati corretti per ciascun salto del programma. In questi casi, i dati di prova devono essere sufficientemente completi da verificare ogni istruzione del programma, come abbiamo già visto.

Grafici e istogrammi

Se usate il computer per analizzare dei dati, è quasi certo che ad un certo punto sentirete la necessità di ottenerne una visualizzazione in forma grafica o di raggrupparli in tabelle di frequenza. I paragrafi che seguono contengono, per l'appunto, la descrizione di una serie di subroutine che consentono di soddisfare questa esigenza.

Per consentire alle subroutine di essere compa-

tibili, è necessario standardizzare alcuni nomi di variabili. Le routine sono state scritte per permettere l'elaborazione di dati fino ad un massimo di 100 valori, contenuti nel vettore V. E' quindi necessario poter avere nel programma principale un'istruzione DIM V(100). Se, ad esempio, i dati devono essere raggruppati in una tabella di frequenza, prima di riprodurli su carta sotto forma di istogramma, la variabile verrà memorizzata nel vettore X e la frequenza nel vettore F. Per la maggior parte degli obiettivi, occorre di solito che la tabella di frequenza con un massimo di quindici intervalli di classificazione venga opportunamente modificata; quindi è necessario che il programma principale abbia un'istruzione DIM contenente X(15), F(15).

Subroutine di raggruppamento in base a frequenza

Prima di realizzare un istogramma o di eseguire altre forme di analisi, può essere necessario raggruppare i singoli valori di dati in intervalli di classificazione, annotando il numero complessivo di valori che rientrano in ciascuno di essi (in altre parole, la frequenza).

La subroutine che permette di raggiungere questo scopo (tabella 9.1) consente di realizzare una tabella di frequenza costituita da 15 intervalli di classificazione. Tutti i dati esclusi per effetto di questo limite vengono stampati con la linea 2100, dopo la quale può far seguito una nuova esecuzione del programma con i parametri di intervallo di classificazione specificati di conseguenza.

Il motivo per cui si tende a realizzare un programma in questo modo è che una routine per l'impostazione completamente automatica dei parametri può cammuffare la presenza di un valore indesiderato che spesso può essere comodo poter ignorare.

Nella tabella 9.2 è stato riportata una subroutine che produce l'uscita stampata di una tabella di frequenza. Dal momento che è vista indipendente dalla subroutine di raggruppamento, l'intervallo di classificazione e i limiti inferiori vengono calcolati dai valori di vettore di X.

```
20 DIM X(15), F(15), V(100)
30 PRINT"SCRIVI IL NUMERO DEGLI ELEMENTI"
40 INPUT N
50 PRINT
60 FOR I=1 TO N
70 PRINT"VALORE DELL'ELEMENTO";I
80 INPUT V(I)
90 NEXT I
100 PRINT
110 GOSUB 2000
120 END
130 :
140 :
150 :
2000 PRINT"SCRIVI L'AMPIEZZA"
2005 PRINT"DELL'INTERVALLO"
2010 INPUT C
2015 PRINT
2020 PRINT"VALORE MINIMO"
2025 PRINT"ACCETTABILE"
2030 INPUT L
2040 FOR I=1 TO N
2050 FOR J=1 TO 15
2060 IF V(I) >= (L+(C*J)) THEN 2090
2070 F(J) = F(J+1)
2080 GOTO 2110
2090 NEXT J
2095 PRINT
2100 PRINT V(I);"E' OLTRE IL"
2105 PRINT "LIMITE STABILITO"
2110 NEXT I
2120 FOR J=1 TO 15
2130 LET X(J) = L + ((J-.5)*C)
2140 NEXT J
2150 RETURN

READY.
```

Tabella 9.1 Routine di raggruppamento in base a frequenza

```
3000 PRINT
3010 LET C = X(2) - X(1)
3020 LET L = X(1) - (.5*C)
3030 PRINT"------------------"
3040 PRINT TAB(4);"X";TAB(14);"F"
3050 PRINT"------------------"
3060 FOR I=1 TO 15
3070 LET B = L + C*(I-1)
3080 PRINT B;TAB(4);"-";TAB(14);F(I)
3090 NEXT I
3100 PRINT"------------------"
3110 RETURN

READY.
```

Tabella 9.2 Routine della tabella di frequenza

• *Problema 1.* **Tabella di frequenza degli alpeggi.**

Scrivete un programma che contenga queste subroutine per l'elaborazione dei dati riportati nella tabella 9.3. Il risultato deve essere una tabella di frequenza degli alpeggi espressi in percentuale.

Provincia	Superficie (in percentuale) degli alpeggi
1	46
2	47
3	63
4	74
5	76
6	26
7	37
8	39
9	35
10	43
11	52
12	59

Tabella 9.3 Dati per provincia

Troverete il programma, che fornisce i risultati riportati nelle tabelle 9.4 e A16.

X	F
20 –	1
30 –	3
40 –	3
50 –	2
60 –	1
70 –	2
80 –	0
90 –	0
100 –	0
110 –	0
120 –	0
130 –	0
140 –	0
150 –	0
160 –	0

Tabella 9.4 Tabella di frequenza per il problema 2

Campionamento da una distribuzione di frequenza

La subroutine che vi descriviamo in questo paragrafo serve a campionare un valore a partire da una distribuzione di frequenza, contenuta nel vettore bidimensionale X. La prima dimensione contiene un valore di variabile, la seconda, la frequenza complessiva in percentuale.

Per permettere alla routine di essere impiegata in modo più generico in più programmi diversi fra loro, è necessario un intervento di standardizzazione del vettore che contiene la distribuzione di frequenza. Innanzitutto è necessario fissare a 10 il numero degli intervalli di classificazione, con il risultato di avere un vettore X di dimensione (10,2). Notate che, per maggiore comodità, gli indici 0 sono stati ignorati. Se una certa distribuzione contiene meno di dieci righe (cioè intervalli di classificazione), gli ingressi definitivi nel vettore saranno identici. Per esempio, i dati da campionare, riportati nella tabella 9.5, sarebbero contenuti nel vettore X(R,I), come indicato nella tabella 9.6.

Variabili	Frequenza cumulativa in percentuale
5	10
10	27
15	42
20	65
25	80
30	100

Tabella 9.5 Dati da campionare

		Indice di colonna, I	
		(,1)	(,2)
Indice di riga, R	(1,)	5	10
	(2,)	10	27
	(3,)	15	42
	(4,)	20	65
	(5,)	25	80
	(6,)	30	100
	(7,)	30	100
	(8,)	30	100
	(9,)	30	100
	(10,)	30	100

Tabella 9.6 Contenuto di X(R,I)

Tutte le distribuzioni utilizzate dal programma principale vengono fissate nello stesso formato (10,2) e, prima dell'ingresso nella subroutine, il vettore X può essere reso uguale ad esse.

Nella figura 9.1 è riportato lo schema a blocchi della subroutine menzionata; nella tabella 9.7 il relativo listato.

Figura 9.1 Schema a blocchi di campionamento

```
900 REM ESEMPIO DI SOTTOPROGRAMMA
910 LET Z = 100*RND(3)
920 FOR R=1 TO 10
930 LET V = X(R,1)
940 IF Z <= X(R,2) THEN 980
950 NEXT R
960 PRINT "ERRORE: IL VALORE CASUALE"
965 PRINT"NON PUO' ESSERE"
968 PRINT"CLASSIFICATO."
970 END
980 RETURN

READY.
```

Tabella 9.7 Routine di campionamento

Il numero casuale generato viene scalato in modo da rientrare fra 0 e 100 (linea 910). Nell'ambito del loop FOR, il vettore X viene controllato riga per riga. Il valore della variabile di riga corrente viene assegnata a V (linea 930) mentre quello del numero casuale Z scalato viene confrontato con la frequenza cumulativa corrente (linea 940). Se Z è maggiore della frequenza, lo stesso procedimento viene ripetuto per la riga successiva (linea 950) e, se eventualmente il valore casuale Z rientra nell'intervallo di classificazione corrente, la subroutine viene abbandonata riportando il valore corrente della variabile V. Se, per errore di impostazione della distribuzione, il valore casuale Z non può essere associato a nessuna riga particolare, si giungerà alle linee 960 e 965 con la conseguente comparsa di un messaggio di errore.

• *Problema 2.* **Subroutine di ingresso di dati.**

Scrivete una subroutine che consenta di introdurre i valori della tabella 9.5 nel vettore bidimensionale D. Fate in modo che possano essere introdotte fino a 10 righe di dati.

Nella tabella A17 troverete una subroutine che consente di soddisfare queste esigenze.

Routine di ingresso protetto

Le istruzioni di ingresso possono provocare spesso gravi problemi agli utenti meno esperti. Se, per esempio, si preme il tasto RETURN in risposta ad un'istruzione di ingresso senza introdurre dati, il Vic accetta quel comando come RETURN nullo e continua il programma, spesso con risultati imprevedibili. La routine che vi proponiamo (tabella 9.8) fà in modo che il RETURN da solo non venga accettato come istruzione, evitando così impostazioni di dati nulle.

La routine fa uso dell'istruzione GET che accetta singoli caratteri dalla tastiera senza attendere il carattere di ritorno e permette l'esame da parte del Vic di qualsiasi ingresso da tastiera. Il suo uso, combinato con quello delle istruzioni di salto condizionale, può essere efficace nella scrittura di programmi che consentono all'utente di scegliere una certa strada fra più alternative di un menu. Poichè

```
5000 REM INPUT PROTETTO
5010 ZI$="": ZJ$=""
5020 PRINT"▨ ▇▌";: FORD =1 TO 50: NEXT
5030 PRINT" ▌";: FOR D=1 TO 50: NEXT
5040 GET ZI$: IF ZI$="" THEN 5020
5050 IF ZI$ <> CHR$(20) THEN 5080
5055 ZL = LEN(ZJ$): IF ZL <1 THEN 5020
5060 ZJ$ = LEFT$(ZJ$,ZL-1)
5070 PRINT ZI$;: GOTO 5020
5080 IF ZI$ = CHR$(13) THEN 5120
5090 PRINT ZI$;
5100 ZJ$ = ZJ$ + ZI$:
5110 GOTO 5020
5120 IF ZJ$ = "" THEN 5020
5130 PRINT
5140 RETURN

READY.
```

Tabella 9.8 Routine di ingresso protetto

l'istruzione GET può accettare anche le risposte nulle, l'utente non ha tempo per rispondere a menochè nel programma non si preveda un loop.

In questo caso si tende di solito ad usare un'istruzione del tipo:

```
200 GET A$: IF A$ = ""THEN 200
```

che forma un loop chiuso che si spezza solo alla pressione di un tasto qualsiasi e, fintantochè è in corso di esecuzione, provoca la mancata visualizzazione del cursore.

Il principio su cui si basa questa subroutine è quello secondo cui una serie di caratteri battuti in risposta ad un loop GET sono concatenati in una stringa in modo da dare l'equivalente di una risposta ad un istruzione INPUT. La linea 5110 stabilisce che la variabile ZI$ dell'istruzione GET e la variabile concatenata ZJ$ siano stringhe nulle. Le linee 5020 e 5030 simulano un cursore lampeggiante, quindi le linee dalla numero 5020 alla numero 5040 formano un loop GET più elaborato, nel senso che la linea 5020 corrisponde a "cursore on", la linea 5030 a "cursore off" e la linea 5040 "GET il carattere, altrimenti cursore on" ecc..

Non appena viene introdotto un carattere, si ha l'abbandono del loop e il test di verifica del carattere.

Il primo test (linea 5050) consiste nel verificare la presenza del tasto di cancellazione (codice di carattere 20). Se il tasto DELETE è stato premuto, viene cancellata la lunghezza della stringa ZJ$ corrente (linea 5055) e se quest'ultima è ridotta ad una stringa nulla, segue la reintroduzione del loop di ingresso, cioè si ha il passaggio del controllo alla linea 5020.

Verificato se ZJ$ contiene caratteri, là linea 5060 provvede ad eliminare quello più a destra mentre la linea 5070 fa sì che il cursore si sposti verso sinistra. Subito dopo si ha il reingresso nel loop GET.

Il secondo test (linea 5080) verifica se il tasto di ritorno (carattere 13) è stato o meno premuto. In caso negativo, viene stampato il carattere accettabile che viene quindi concatenato a ZJ$ in corrispondenza delle linee 5090 e 5100 prima che 5110 restituisca il controllo al loop GET.

Quando viene premuto il tasto RETURN, la linea 5080 provoca la prosecuzione dell'esecuzione con la linea 5120 in cui viene controllato lo stato della stringa ZJ$. Se quest'ultima è ancora una stringa nulla, la pressione del tasto RETURN viene considerata inaccettabile si ha un nuovo ingresso nel loop GET. Se invece ZJ$ contiene una risposta, la pressione del tasto RETURN viene convalidata e acquisita e ad essa fà seguito l'esecuzione dell'istruzione PRINT finale che conclude l'azione dei punti e virgola delle istruzioni di stampa precedenti.

Grafici ad alta risoluzione

Il computer consente di creare caratteri definiti dall'utente semplicemente selezionando il modo di traccia per singoli punti (pixel) anzichè per caratteri predefiniti. Ogni carattere viene definito partendo da un vettore (matrice) di 8 x 8 pixel e grazie a questa alta risoluzione, è possibile creare grafici di ottima qualità. Per poter manipolare i caratteri con questo obiettivo, è necessario prendere quelli contenuti nella ROM e spostarli nella RAM dove possono essere opportunamente corretti. Ciò significa dover modificare anche i punti nella memoria, in modo che il computer possa essere diretto verso la

parte di RAM interessata.

A questo proposito è disponibile una cassetta ad alta risoluzione destinata ai Vic che dispongono di una versione del BASIC ampliata con una serie di comandi speciali per l'esecuzione di grafici ad alta risoluzione. In ogni caso, chiunque abbia ùn VIC con una memoria di espansione di 3 o più Kbyte è in grado di ottenere ottime prestazioni in questo senso semplicemente utilizzando le subroutine riportate qui di seguito.

L'esame dettagliato di ciò che occorre in questo caso esula dagli scopi di questo corso, tuttavia tutte le routine necessarie possono essere incorporate nei programmi scritti dall'utente nel modo seguente:

```
1000 REM INIZIALIZZAZIONE
1001 REM PER L'ALTA RISOLUZIONE
1005 PRINT"■":REM TASTI <CTRL> & 1
1010 PRINT"]": POKE 36879,79
1020 IF PEEK(36869)=206 THEN 1080
1030 POKE 36869,206: POKE 36867,PEEK(36867) OR 128
1040 POKE 55,0: POKE 56,24: POKE 51,0: POKE 52,24
1050 RUN 1060
1060 S=32768: T=6144: PRINT"]INIZIALIZZAZIONE"
1070 FOR I=0 TO 255*8+7: POKE I+T,PEEK(I+S): NEXT
1080 REM FINE DEL PROCEDIMENTO
```

Tabella 9.9 Inizializzazione ad alta risoluzione

Per scrivere un programma che incorpori grafici ad alta risoluzione, è necessaria l'applicazione di due routine, la prima delle quali è riportata nella tabella 9.9. Questa deve essere inserita sempre all'inizio del programma ed ha due importanti funzioni: modifica i puntatori e stabilisce le caratteristiche dello schermo e della cornice e il colore dei caratteri (rosso, giallo e nero), mentre i punti vengono sempre tracciati in bianco. La seconda routine, riportata nella tabella 9.10 esegue una certa traccia sullo schermo, a partire dall'angolo in alto a sinistra, ad uno specifico pixel usando X e Y come coordinate. Questa seconda routine può essere usata anche come subroutine. Un esempio di impiego di entrambe le routine è riportato nel prossimo paragrafo.

```
8000 REM PLOT
8010 X%=X/8:Y%=Y/8:P=X%+Y%*22+4096:Q=PEEK(P):IFQ>=128THEN8060
8030 CN=CN+1:S=6144+(127+CN)*8:T=6144+Q*8
8040 FORI=0TO7:POKES+I,PEEK(T+I):NEXT
8050 Q=127+CN:POKEP,Q
8060 C=6144+Q*8+(YAND7):POKEC,(2↑(7-(XAND7)))ORPEEK(C)
8080 RETURN
```

Tabella 9.10 Subroutine per la traccia di grafici

Esempio di traccia grafica ad alta risoluzione

Il programma che esamineremo ora disegna il grafico dell'andamento di borsa dei titoli azionari di un'azienda dolciaria e del prezzo del cacao. Poichè nello spazio di un carattere è possibile tracciare fino a 8 punti orizzontali, una scala particolarmente efficace nelle applicazioni gestionali è di due pixel a settimana. Ciò consente di usare 104 pixel (2 x 52 settimane) che possono essere scalati con i caratteri grafici tradizionali in tredici periodi quadrisettimanali.

Questo principio, secondo cui l'anno può essere immaginato costituito da 13 periodi di quattro settimane, viene usato spesso nelle applicazioni gestio-

Tabella 9.11 Esempio di traccia di un grafico ad alta risoluzione (funziona solo con 8K aggiuntive di memoria).

```
2000 POKE36879,8:REM **** SOTTOPROGRAMMA ****
2010 PRINT"▓▓CAPITALE";TAB(15);"CACAO","PREZZO";TAB(15);"PREZZO▓▓▓"
2025 READLB:LT=LB+100:READRB:RT=RB+100
2030 FORI=0TO10:X=LT-10*I:Y=RT-10*I
2040 X$=RIGHT$("  "+STR$(X),3):Y$=RIGHT$("  "+STR$(Y),3):PRINTX$;TAB(17);Y$:NEXT
2058 REM ASSI VERTICALI
2060 FORI=5TO15:POKE4096+22*I+3,115:POKE4096+22*I+16,115:NEXT
2078 REM INTERSEZ. E ASSE ORIZZONTALE
2080 POKE4096+15*22+3,91:FORI=4TO15:POKE4096+15*22+I,114:NEXT
2100 PRINT"▓";TAB(3);"1234567890123"
3000 REM PRIMO ISTOGRAMMA
3010 FORX=28TO132
3020 READD:IFD=0THEN4000
3030 Y=INT((LT+55-D)*8/10):GOSUB8000:NEXT
4000 REM SECONDO GRAFICO
4010 FORX=28TO132:READD:IFD=0THEN4060
4030 Y=INT((RT+55-D)*8/10):GOSUB8000:NEXT
4060 GOTO9000:REM FINE SECONDO GRAFICO
6000 DATA60,100
6010 REM PRIMO GRAFICO
6020 DATA100,102,105,108,110,115,112,118
6022 DATA120,122,118,120,120,124,126,130
6024 DATA128,126,128,126,125,124,126,122
6026 DATA122,126,128,130,132,136,140,142
6028 DATA144,142,139,138,136,137,135,138
6030 DATA136,134,132,132,130,128,130,134
6032 DATA0
6100 REM SECONDO GRAFICO
6110 DATA102,104,110,106,108,110,110,108
6112 DATA110,112,113,115,118,114,116,118
6114 DATA120,122,124,120,118,116,114,110
6116 DATA110,112,116,118,120,122,124,120
6118 DATA120,118,118,116,114,113,116,115
6120 DATA114,110,112,108,106,110,114,116
6122 DATA0

9000 REM FINE
9010 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓PREMI SPAZIO":WAIT203,32:POKE36879,27:PRINT"▓▓":
END
```

nali e finanziarie che richiedono analisi particolari di dati.

La scelta di rappresentare un anno con 104 pixel lascia sufficiente spazio sullo schermo per inserire notazioni e didascalie.

Il prezzo di borsa dei titoli della società presa in considerazione va collocato sull'asse sinistro, quello del prezzo del cacao sull'asse destro. Nella tabella 9.11 è riportata la routine utente che va usata tenendo sott'occhio le tabelle 9.9 e 9.10. Le linee dalla numero 2000 alla numero 2100 fissano le intestazioni e le scale, le linee dalla 3000 alla 3030 tracciano le variazioni di prezzo della quota azionaria e le linee dalla 4000 alla 4060 i prezzi del cacao. I dati sono invece riportati nella parte di programma che va dalla linea 6000 alla linea 6122.

La maggior parte delle intestazioni e delle scale è determinata dalle istruzioni PRINT, mentre qui l'inizio di tutte le scale verticali viene letto in LB ed RB nella linea 2025. Viene quindi calcolata la scala più appropriata.

Per tracciare le variazioni dell'andamento in borsa della società in questione (dalla linea 3000 alla linea 3030), viene predisposto un loop in grado di eseguire una traccia orizzontale da pixel 28 al pixel 132 e viene comunque abbandonato se il valore del dato letto, D, è zero. Così con il trascorrere dell'anno, solo le istruzioni relative ai dati devono essere ampliate e chiuse con uno zero. Il valore D viene quindi messo in scala nel valore di pixel opportuno nella linea 3030. Una volta che il valore Y è stato calcolato, viene chiamata la subroutine di traccia (8000).

L'esecuzione del grafico relativo alle variazioni di prezzo del cacao presenta uno schema e una codifica simile a quella della quota azionaria. Poichè la routine iniziale sposta i puntatori in quella direzione, al completamento del grafico non comparirà il cursore. Quindi una piccola routine finale (9000), che chiami un'istruzione GET per chiudere il programma, è indispensabile.

Per evitare che la visualizzazione sia disturbata dal ritorno del cursore, il messaggio da stampare (nella linea 9010) viene fatto precedere da una serie di caratteri "cursore giù", ciascuno dei quali sposta il cursore in giù di una linea dello schermo.

Il Vic in italiano

Come visualizzare i messaggi di errore tradotti in italiano.

QUANDO si dà il RUN ad un programma non ancora collaudato, si entra, inconsciamente, in una fase di estrema tensione dove il semplice SYNTAX ERROR può causare una crisi depressiva che, col ripetersi del fenomeno (semmai con l'aiuto di qualche ILLEGAL QUANTITY) può diventare estremamente pericolosa per l'incolumità dell'"aguzzino".

Per evitare guai di qualsiasi natura, si può prendere uno dei seguenti provvedimenti:
1/ creare un programma che corregga i programmi;
2/ darsi all'ippica;
3/ rendere meno "drastici" e più confidenziali i messaggi che il computer ci elargisce con tanta munificenza, facendo in modo che appaiano in italiano.

Dato che la prima ipotesi è alquanto utopistica e che la seconda non risolve il problema, si deve prendere in considerazione la terza.

La soluzione

Siccome qualcosa di questo genere sarebbe stata difficile da realizzare in Basic (avrebbe occupato molta memoria e non sarebbe stata veloce), ho pensato, come unica alternativa, di ricorrere al Linguaggio Macchina. Mi sono messo al lavoro e sfogliando alcuni validi testi (tra i quali M&P n. 30 pag. 28), ho notato che i progettisti del Vic, hanno messo in alcune locazioni RAM (guarda caso) dei puntatori che forniscono l'indirizzo delle più importanti routine del Basic; uno di questi "dice" (ricordo che un puntatore è composto da due byte consecutivi) al Basic dove si trova la routine che stampa gli errori. Le locazioni interessate sono la $0300 e la $0301 (768-769) che, contenendo rispettivamente $3A e $C4, indirizzano alla routine posta a partire da $C43A. Prima di continuare, desidero precisare che se è presente una espansione ROM in $A000 (per esempio, l'espansione grafica), tutto quanto esposto sopra non vale sempre, ma la routine che si sta per presentare funziona lo stesso. Ciò premesso, passo a descrivere le operazioni svolte dalla sopracitata routine: mediante il codice d'errore riportato dalla routine che controlla l'esatta sintassi di una linea Basic, individua il puntatore del messaggio da stampare, chiude tutti i file aperti e i canali di comunicazione, stampa il messaggio relativo al codice d'errore e, quindi, ridà il controllo al Basic. La routine "originale" dalla quale ho ricavato quella "italianizzata", va a leggere una *word table* residente in ROM, indirizzata da puntatori contenuti anch'essi in ROM.

Le precauzioni

Il lavoro da fare, quindi, consiste nell'adattare la routine alle nuove esigenze, creare una *word table* in italiano con annessi puntatori e scrivere il tutto in una zona RAM non dimenticando di modificare il puntatore $0300 - $0301, in modo che indirizzi alla nuova routine (fig. 1).

Il risultato

Dopo aver superato le difficoltà "concettuali", è ora di varcare la soglia della realizzazione pratica; per rendere universale (o quasi) questa routine, ho scelto di spostare la mappa video a $1000 (4096) e accordare il tut-

Per caricare il programma 1

1/ Vic senza espansioni o con espansione di 3K eseguire:
   a) POKE 36869,192: POKE 648,16: POKE 56,31: POKE 55,255
   b) premere RUN/STOP insieme a RESTORE
   c) POKE 43,176: POKE 44,18: POKE 4783,0: NEW
2/ Per Vic con espansioni di 8 o 16 K
   x) eseguire solo la fase c) precedente.

```
1 REM*******************************************
2 REM*                                         *
3 REM* IL  VIC  IN...  ITALIANO  (PROGRAMMA 1)  *
4 REM*                                         *
5 REM*  BY PASQUALE D'ANDRETI TEL.0823/981216   *
6 REM*  VERSIONE CONSIGLIATA PER VIC INESPANSI  *
7 REM*******************************************
8 :
10 PRINT"ATTENDI":L=4608
20 READA$:IFA$="X"THEN50
30 A=ASC(A$)-48:B=ASC(RIGHT$(A$,1))-48:N=B+(B>9)*7+(16*((A>9)*7+A))
40 POKEL,N:L=L+1:GOTO20
50 PRINT"FINITO":POKE768,0:POKE769,18:END
51 REM*
52 REM*ROUTINE STAMPA ERRORI
53 REM*
60 DATA8A,0A,AA,BD,6D,12,85,22,BD,6E,12,85,23,4C,4A,C4
101 REM*
102 REM*MESSAGGI IN ITALIANO
103 REM*
110 DATA53,43,52,49,56,49,20,42,45,4E,45,2C,20,43,52,45
120 DATA54,49,4E,4F,A1,43,49,20,56,55,4F,4C,45,20,49,4C
130 DATA20,46,4F,52,2C,20,53,43,45,4D,4F,A1,4D,45,54,54
140 DATA49,20,49,4C,20,47,4F,53,55,42,A1,43,27,45,27,20
150 DATA55,4E,41,20,4C,49,4E,45,41,20,46,41,4E,54,41,53
160 DATA4D,C1,52,49,56,45,44,49,20,49,20,44,41,54,C1
401 REM*
402 REM*PUNTATORI
403 REM*
410 DATA9E,C1,AC,C1,B5,C1,C2,C1,D0,C1,E2,C1,F0,C1,FF,C1
420 DATA10,C2,25,12,10,12,3C,12,62,12,5A,C2,6A,C2,72,C2
430 DATA4B,12,90,C2,9D,C2,AA,C2,BA,C2,C8,C2,D5,C2,E4,C2
440 DATAED,C2,00,C3,2D,C3,1E,C3,24,C2,83,C3
441 REM*
442 REM*FINE CODICI
443 REM*
450 DATAX
```

```
1 REM***********************************
2 REM*                                 *
3 REM*  IL VIC IN ITALIANO.PROGRAMMA 2 *
4 REM*                                 *
9 REM***********************************
10 PRINT"ATTENDI":L=4608
20 READA$:IFA$="X"THEN50
30 A=ASC(A$)-48:B=ASC(RIGHT$(A$,1))-48:N=B+(B>9)*7+(16*((A>9)*7+A))
40 POKEL,N:L=L+1:GOTO20
50 PRINT"FINITO":POKE768,0:POKE769,18:END
51 REM*
52 REM*ROUTINE STAMPA ERRORI
53 REM*
60 DATA86,FB,A9,41,A0,12,20,1E,CB,A6,FB,8A,0A,AA,BD,27
```

```
B*
   PC  SR AC XR YR SP
.;603E 33 00 63 00 F6
.
.,  1200 STX $FB
.,  1202 LDA #$41
.,  1204 LDY #$12
.,  1206 JSR $CB1E
.,  1209 LDX $FB
.,  120B TXA
.,  120C ASL
.,  120D TAX
.,  120E LDA $1427,X
.,  1211 STA $22
.,  1213 LDA $1428,X
.,  1216 STA $23
.,  1218 JSR $FFCC
.,  121B LDA #$00
.,  121D STA $13
.,  121F JSR $CAD7
.,  1222 JSR $CB45
.,  1225 LDY #$00
.,  1227 LDA ($22),Y
.,  1229 PHA
.,  122A AND #$7F
.,  122C JSR $CB47
.,  122F INY
.,  1230 PLA
.,  1231 BPL $1227
.,  1233 JSR $C67A
.,  1236 LDY $3A
.,  1238 INY
.,  1239 BEQ $123E
.,  123B JSR $DDC2
.,  123E JMP $C474
```

Commento

$1200-$1206: salva il registro X nella locazione $FB e stampa "ERRORE";
$1209-$120D: carica il registro X nell'accumulatore e lo moltiplica per 2, in modo che sommato alla locazione d'inizio dei puntatori dei messaggi, ne trovi l'Xesimo, il quale, a sua volta, conterrà l'indirizzo dell'Xesimo messaggio;
$120E-$1216: carica il puntatore trovato in due locazioni di pagina zero ($22-$23), in modo che si possa ricorrere all'indirizzamento indicizzato indiretto;
$1218: chiude tutti i canali e i file e mette in default l'I/O (tastiera e video);
$121B-$1222: prepara la stampa su video;
$1225-$1231: esegue un loop in cui, ad ogni ciclo, carica la Yesima lettera del messaggio puntato da $22-$23, la stampa e, nel caso che non fosse l'ultima (il codice ASCII dell'ultima lettera è maggiorato di $80 rispetto a quello normale), incrementa Y di 1 e ripete il ciclo;
$1233-$1239: risistema le variabili e controlla se l'errore è generato da una linea di programma o da un comando diretto;
$123B: se si è verificato il primo caso scrive "IN" seguito dal numero della linea "incriminata";
$123E: cede il controllo al Basic.

digitare POKE 36869,192: POKE 648,16: POKE 56,31: POKE 55,255 e, senza preoccuparsi di ciò che appare sullo schermo, RUN/STOP & RESTORE oltre, a quanto detto prima. Tutto questo lo si deve fare SEMPRE, cioè sia prima dell'inserimento da tastiera, sia prima della lettura da nastro o disco. Dopo aver trascritto esattamente il programma facendo attenzione soprattutto ai codici esadecimali (bisogna metterci un po' di buona volontà), dare il RUN ed attendere fin quando non compare la scritta "FINITO"; cancellare il programma con NEW per (finalmente!) consolarsi della fatica sostenuta (se durante l'inserimento ci si scoraggia, si pensi a me, che, oltre che scrivere ho dovuto anche calcolare e controllare tutti quei codici, uno per uno). Dando per scontato che il "collaudo" possa essere fatto anche senza le mie indicazioni (!), non mi rimane che raccomandare che non si "sporchi" l'area di memoria occupata dalla routine con delle POKE e che non si sposti la mappa video e l'inizio del Basic.

Per caricare il programma 2

1/ Vic senza espansioni
  x) eseguire fase a) e fase b) programma 1
  b) POKE 43,100: POKE 44,20: POKE 5219,0: NEW
2/ Vic con espansione da 8 oppure 16K
  eseguire solo fase b) precedente.

Conclusioni

Dopo tutto questo lavoro, sarebbe ingiusto terminare il discorso senza esprimere delle opinioni sia pur di carattere non del tutto tecnico, perciò, vorrei fare osservare l'importanza, a volte determinante, dell'impiego di subroutine e di salti indiretti nei programmi in L.M.. Si noti, infatti, che i progettisti del Vic non solo hanno creato un hardware estremamente flessibile (user port e connettore per espansioni ne sono il segno), ma anche un software di base completo, sofisticato e, nello stesso tempo, facile da modificare secondo le esigenze individuali, proprio per la presenza di numerose subroutine e salti indiretti. ■

Le funzioni logiche

Un programma utile per lo studio delle tabelle della verità.

QUANDO un appassionato di personal computer, decide di approfondire lo studio del proprio apparecchio, si trova immediatamente di fronte a circuiti digitali che simulano, elettricamente, alcune "porte" logiche elementari.

L'esame di queste ultime, esula dallo scopo del presente articolo. Riportiamo, tuttavia, gli schemi grafici e le relative tavole della verità, interessate dal programma pubblicato.

Che cosa fa il programma

Dopo aver scelto quale funzione logica considerare, è necessario digitare, in decimale, i due valori da sottoporre al confronto. Compariranno, immediatamente, incolonnati, i due valori stessi tradotti in binario ed il risultato, sia in binario che in decimale, dell'operazione logica compiuta.

I valori accettati in INPUT devono, ovviamente, essere interi e compresi tra 0 e 255. Digitando un valore esterno all'intervallo citato si ritorna al menu principale.

Il programma può essere usato, su *qualsiasi* computer Commodore, senza comprenderne il funzionamento. I più interessati potranno però studiare le semplici subroutine (720-990) in modo da avere un suggerimento su come operare nel caso si presenti la necessità di realizzare funzioni logiche in altri programmi. ▪

AND OR NAND NOR
OR ESCL. NOR ESC.
INVERTER
QUALSIASI COMPUTER

```
210 PRINT"■CHE COSA VUOI FARE ?"
220 PRINT"■1) FUNZIONE AND
230 PRINT"■2) FUNZIONE OR
240 PRINT"■3) FUNZIONE NAND
250 PRINT"■4) FUNZIONE NOR
260 PRINT"■5) FUNZIONE OR ESCL.
270 PRINT"■6) FUNZIONE NOR ESCL.
280 PRINT"■7) FUNZIONE INVERTER
290 PRINT"■0) FINE LAVORO
300 GETA$:IFA$=""THEN300
310 IFA$="1"THENY$="■AND■ ":GOSUB410:GOTO210
320 IFA$="2"THEN Y$="■OR■ ":GOSUB440:GOTO210
330 IFA$="3"THEN Y$="■NAND■ ":GOSUB470:GOTO210
340 IFA$="4"THEN Y$="■NOR■ ":GOSUB500:GOTO210
350 IFA$="5"THEN Y$="■OR EX■ ":GOSUB530:GOTO210
360 IFA$="6"THEN Y$="■NOR EX■ ":GOSUB560:GOTO210
370 IFA$="7"THEN Y$="■INVER.■ ":GOSUB590:GOTO210
380 IFA$="0"THEN END
390 GOTO210:DE SIMONE SOFTWARE 83
400 REM *** FUNZIONE AND ***
410 X7=0: GOSUB630:IF X7=1 THEN RETURN
420 GOSUB700:GOSUB730: GOTO410
430 REM *** FUNZIONE OR ***
440 X7=0: GOSUB630:IF X7=1 THEN RETURN
450 GOSUB700:GOSUB770: GOTO440
460 REM *** FUNZIONE NAND ***
470 X7=0: GOSUB630:IF X7=1 THEN RETURN
480 GOSUB700:GOSUB810: GOTO470
490 REM *** FUNZIONE NOR ***
500 X7=0: GOSUB630:IF X7=1 THEN RETURN
510 GOSUB700:GOSUB860: GOTO500
520 REM *** FUNZIONE OR ESCLUSIVO ***
530 X7=0: GOSUB630:IF X7=1 THEN RETURN
540 GOSUB700:GOSUB910: GOTO530
550 REM *** FUNZIONE NOR ESCLUSIVO ***
560 X7=0: GOSUB630:IF X7=1 THEN RETURN
570 GOSUB700:GOSUB960: GOTO560
580 REM *** FUNZIONE INVERTER ***
590 PRINTY$"NUM. DEC.";:INPUT X
600 IFX<0 OR X>255 OR X <> INT(X)THEN X7=1:RETURN
610 X8=X:GOSUB1010:GOSUB1070:GOSUB1090:GOTO590
```

Tavole della verità

FUNZIONE	SIMBOLO GRAFICO	INGRESSI		USCITA
		A	B	U
AND		0 1 0 1	0 0 1 1	0 0 0 1
OR		0 1 0 1	0 0 1 1	0 1 1 1
NAND		0 1 0 1	0 0 1 1	1 1 1 0
NOR		0 1 0 1	0 0 1 1	1 0 0 0
OR ESCLUSIVO		0 1 0 1	0 0 1 1	0 1 1 0
NOR ESCLUSIVO		0 1 0 1	0 0 1 1	1 0 0 1
INVERTER		0 1	/ /	1 0

DAL N°1: COM

Mai un grande perso

Quest'anno, fatti un regalo intelligente: un computer dalle caratteristiche incredibili. Vediamole.

1. Commodore 64 è potente, sofisticato, professionale.

2. Ha una vastissima gamma di programmi già pronti, lo usi nella professione, a casa, a scuola, nella ricerca scientifica, con facilità e totale affidabilità.

3. Ha un'incredibile memoria (64 K), un sintetizzatore sonoro professionale, produce effetti tridimensionali.

4. Ti diverti perchè è anche un sofisticato videogioco.

5. Con Commodore 64 entri nel futuro, tasto dopo tasto.

6. Commodore 64 oggi lo puoi avere a prezzo davvero speciale:

MODORE 64

nal è costato così poco

approfittane però perchè sta andando a ruba,
e chi primo arriva...
Vieni a un punto vendita Commodore:
ti aspetta una bella sorpresa.

Commodore Italiana S.p.A.
Via F.lli Gracchi 48 - Cinisello Balsamo (MI)
Tel. 02/6125651-6123253

commodore
COMPUTER

DAL N°1:

Perchè accontentarsi di un videogame?

Speciale, specialissimo!
Invece dei soliti videogiochi prova VIC 20, e guarda quante cose fa in più!

1. VIC 20 ha una valanga di videogiochi, uno più bello dell'altro, uno più nuovo dell'altro.
2. Ma VIC 20 è un computer e fa molto di più.
3. Lo usi per la scuola, o per la casa, o per la professione. Ci sono, pronti pronti, un mucchio di programmi. Metti le cassette e via con cose utili.
4. Puoi imparare il BASIC, la lingua del futuro (ed è facile facile imparare a programmare).
5. Nel mondo sono stati venduti più di un milione di VIC, a gente sveglia, quelli del 2000.

```
620 REM *** INPUT DATI ***
630 PRINTY$"1° NUM. DEC.";:INPUT X
640 IFX<0 OR X>255 OR X <> INT(X)THEN X7=1:RETURN
650 X8=X:GOSUB1010:X1$=X0$
660 INPUT"2° NUM. DEC.";X
670 IFX<0 OR X>255 OR X <> INT(X)THEN X7=1:RETURN
680 X9=X:GOSUB1010:X2$=X0$:RETURN
690 REM *** VISUALIZZAZIONE VALORI BINARI ***
700 PRINT"■ ";:FORI=8TO1STEP-1:PRINTMID$(X1$,I,1)" ";:NEXT:PRINT "■"X8
710 PRINT" ";:FORI=8TO1STEP-1:PRINTMID$(X2$,I,1)" ";:NEXT:PRINT "■"X9:RETURN
720 REM *** CALCOLO AND ***
730 X3$="":X1=0:PRINT"■";:FORI=8TO1STEP-1
740 X=VAL(MID$(X1$,I,1)) AND VAL(MID$(X2$,I,1)):X1=X1+X*2↑(I-1)
750 PRINT STR$(X);:NEXT:PRINT"■"X1:RETURN
760 REM *** CALCOLO OR ***
770 X3$="":X1=0:PRINT"■";:FORI=8TO1STEP-1
780 X=VAL(MID$(X1$,I,1)) OR VAL(MID$(X2$,I,1)):X1=X1+X*2↑(I-1)
790 PRINT STR$(X);:NEXT:PRINT"■"X1:RETURN
800 REM *** CALCOLO NAND ***
810 X3$="":X1=0:PRINT"■";:FORI=8TO1STEP-1:REM ING. DE SIMONE
820 X=1:IF (VAL(MID$(X1$,I,1))=1) AND (VAL(MID$(X2$,I,1))=1) THENX=0
830 X1=X1+X*2↑(I-1):REM TEL.039/464446
840 PRINT STR$(X);:NEXT:PRINT"■"X1:RETURN
850 REM *** CALCOLO NOR ***
860 X3$="":X1=0:PRINT"■";:FORI=8TO1STEP-1
870 X=0:IF (VAL(MID$(X1$,I,1))=0) AND (VAL(MID$(X2$,I,1))=0) THENX=1
880 X1=X1+X*2↑(I-1)
890 PRINT STR$(X);:NEXT:PRINT"■"X1:RETURN
900 REM *** CALCOLO OR ESCLUSIVO ***
910 X3$="":X1=0:PRINT"■";:FORI=8TO1STEP-1
920 X=1:IF VAL(MID$(X1$,I,1)) = VAL(MID$(X2$,I,1)) THENX=0
930 X1=X1+X*2↑(I-1)
940 PRINT STR$(X);:NEXT:PRINT"■"X1:RETURN
950 REM *** CALCOLO NOR ESCLUSIVO ***
960 X3$="":X1=0:PRINT"■";:FORI=8TO1STEP-1
970 X=0:IF VAL(MID$(X1$,I,1)) = VAL(MID$(X2$,I,1)) THENX=1
980 X1=X1+X*2↑(I-1)
990 PRINT STR$(X);:NEXT:PRINT"■"X1:RETURN
1000 REM *** TRADUZIONE IN BINARIO ***
1010 X0$="":X1=X
1020 FOR I=7 TO 0 STEP-1
1030 X1=X1/2:X2=INT(X1):IFX1<>X2THENX0$=X0$+"1":X1=X2:GOTO1050
1040 X0$=X0$+"0"
1050 NEXTI:RETURN
1060 REM *** VISUALIZZ. INVERETER ***
1070 PRINT"■ ";:FORI=8TO1STEP-1:PRINTMID$(X0$,I,1)" ";:NEXT:PRINT "■"X8:RETURN
1080 REM *** CALCOLO INVERTER ***
1090 X3$="":X1=0:PRINT"■";:FORI=8TO1STEP-1
1100 X=1:IF VAL(MID$(X0$,I,1)) = 1THENX=0
1110 X1=X1+X*2↑(I-1)
1120 PRINT STR$(X);:NEXT:PRINT"■"X1:RETURN
1130 REM REGOLO "_-_-_-_-_-/_-_-_-_-_-/_-_-_-_-_-/
```

Assembler per tutti

di **Alessandro de Simone**

PRIMA di continuare il discorso sul set di istruzioni del 6502, vediamo di chiarire alcuni concetti già accennati sul numero scorso.

1/ Il calcolatore "ragiona" solo in binario puro, tratta cioè solo gruppi di otto stati di tensione elettrica alla volta alta o bassa, detti BIT.

2/ Per semplicità (vedi fig. 2 n. 1), il dato formato da otto BIT (detto parola o Byte), viene "spezzato" in due da quattro Bit (detti ciascuno Nibble) e "tradotto" in esadecimale, al solo scopo di rendere semplice la vita al programmatore.

3/ Per semplificare ancora di più la stesura di un programma in L.M., si ricorre spesso ad un linguaggio detto "Assembler" che, utilizzando gruppi di lettere derivate dalle iniziali delle parole inglesi, che indicano la funzione dell'istruzione, ed incolonnati uno dopo l'altro, consente una relativa facilità nell'individuare gli eventuali errori o nell'apportare modifiche (fig. 3 n. 1).

```
  LM  A9 00 8D 00 80 60
Ass.@$8000= VIDEO :  la locazione 8000 (esadecimale) è definita
                     come "video";
    LDA# #0:         carica l'accumulatore con il dato che
                     segue immediatamente (cioè zero);
    STA; VIDEO :     trasferisce il valore dell'accumulatore
                     nella locazione definita all'inizio come
                     VIDEO, cioè 8000;
    RTS :            return.
```

Come si può notare nel linguaggio Assembler, non è necessario ricordare a memoria tutte le istruzioni del 6502 sotto forma di coppie di valori esadecimale nè tantomeno indicare volta per volta certe locazioni di memoria con l'indirizzo in esadecimale.

D'ora in poi scriveremo i programmi in L.M., da inserire nel PET tramite il monitor TIM, e nel VIC e nel 64 con i "simulatori" pubblicati nei numeri 5 e 6, con a fianco una "traduzione" in Assembler. Dobbiamo ricordare, infatti, che ancor più che nel caso del BASIC, non esiste un Assembler universale, ma vi sono in commercio diversi tipi di Assembler, che differiscono l'uno dall'altro in alcuni particolari.

Quello pubblicato sul fascicolo n.6, è un programma che consente di compilare (tradurre da linguaggio mnemonico in linguaggio macchina) in Assembler, corredato da istruzioni, che gira su tutti i tipi di PET (2000-3000-4000 e 8000), anche vecchie ROM sia pure da 8K, oltre che su Vic 20 e Commodore 64, e visualizza il programma assemblato su video e su stampante.

Il programma suddetto non è indispensabile, nè lo sarà per le puntate future, al fine di seguire i programmi che verranno descritti su Commodore Computer Club. Potrà, comunque, essere utilissimo quale compendio per i lettori che vorranno stendere programmi per conto proprio o per scrivere quei programmi pubblicati da altre riviste che non riportano anche il così detto "codice oggetto", cioè il programma scritto direttamente in L.M.

Calcolo esadecimale, decimale, binario

Continuiamo ora il discorso sull' L.M., trattando la corrispondenza esistente tra numerazione esadecimale e decimale. Per esempio, per noi il numero 183 significa un numero di oggetti pari alla somma di un centinaio, otto decine e tre unità. Volendo esprimere questo numero come un insieme di potenze di dieci, noi scriviamo:

183 = 1 x 10↑2 +8 x 10↑1 + 3 x 10↑0

Si ricordi che qualsiasi numero, tranne 0, elevato alla potenza nulla, fornisce come risultato il numero 1.

Inoltre, si ricordi che 10↑n è uguale a 10 x 10 x...10 n volte. Per esempio:

10↑4 = 10 x 10 x 10 x 10 = 10.000.

Qualsiasi numero, pertanto, viene da noi rappresentato come la somma di potenze di dieci, decrescenti (2, 1, 0 nel nostro caso, perchè 183 è composto di tre cifre) ciascuna moltiplicata per un fattore che è una delle cifre del numero considerato.

Altri esempi:

Potenza di 10	3	2	1	0	-1	-2	Significato
Valore	8	6	4	8			8x10↑3+6x10↑2+4x10↑1+8x10↑0
			4	2			4x10↑1+2x10↑0
			1	8	,1		1x10↑1+8x10↑0+1x10↑-1
			1	0	,0	2	1x10↑1+0x10↑0+0x10↑-1+2x10↑-2

Tale sistema di numerazione è stato adottato dall'uomo perchè probabilmente quando scoprì i numeri si servì del metodo più semplice di cui potesse disporre: le dita delle mani.

Un calcolatore, e quindi anche un Commodore, ha a disposizione soltanto uno stato alto o basso, a seconda se in un particolare punto del circuito elettrico vi è tensione o meno. Indicando lo stato alto di tensione con 1 e quello basso con 0, il computer dispone di un sistema che ha appena due simboli, e prende appunto il nome di "SISTEMA BINARIO".

Una cifra, in tale sistema, prende il nome di bit, ed una quantità qualunque deve venire espressa come potenza di due.

Concettualmente i due sistemi di numerazione, decimale e binario, sono identici, solo che quello decimale ha a disposizione 10 simboli, quello binario soltanto due, e perciò, per indicare una stessa quantità, saremo costretti ad usare più simboli, ricorrendo al binario, di quanti ne occorrano usando il decimale.

Esempio:

Potenza di 2	3 2 1 0	Significato
Valore	1 1 0	1x2↑2+1x2↑1+0x2↑0 = 5 (dec)
	1 1	1x2↑1+1x2↑0 = 3 (dec)
	1 1 1 1	1x2↑3+1x2↑2+1x2↑0 = 15 (dec)
	0	0x2↑0 = 0

Come abbiamo visto, per rappresentare la quantità quindici, sono sufficienti due simboli (1 e 5) nel sistema decimale e ben quattro (1 1 1 1) in quello binario.

Per quantità non molto più grandi (dell'ordine del milione) c'è bisogno di decine di simboli binari contro i sette del decimale.

Come memorizzarli, pertanto, in un calcolatore senza spreco di memoria? Il metodo è relativamente semplice se si ricorre al sistema esadecimale. Dalla figura 1 della scorsa puntata, si può immaginare facilmente che una certa quantità sarà rappresentata da un numero di simboli esadecimali, inferiore a quello decimale a sua volta nettamente inferiore al binario puro. Poichè la memoria di un calcolatore è una successione di gruppi di otto bit ciascuno, noi potremmo considerare, per semplicità, ciascun byte come se fosse formato da due valori esadecimali, benchè, nella realtà, il valore sia "scritto" in binario puro (fig. 2 num. prec.). Si presenta ora un altro problema nel progetto di un calcolatore, e cioè come individuare, tra le tante, una certa locazione di memoria. Ad ogni byte si assegna un indirizzo che altro non è se non un gruppo di sedici bit.

INDIRIZZO HA	INDIRIZZO LA	DATO		TRAD. ESA. IND.	TRAD. ESA. DATO
0000 0000	0000 0000	0100 0000	= 0	0000	40
0000 0000	0000 0001	0100 0010	=	0001	42
0000 0000	0000 0010	1111 1111	=	0002	FF
....	....	....	=	....	..
1010 0010	0000 1100	1100 0110	=	A20C	C7

In tale modo si possono indirizzare 2↑16=65536 locazioni di memoria. Con un numero inferiore di bit (ad esempio otto), si possono indirizzare solamente 2↑8 = 256 locazioni di memoria, insufficienti per un versatile uso di un computer.

Da notare che, in generale, i microprocessori ad otto bit (6502, 8080, Z80, ecc.), trattano come dati gruppi di otto bit, ed hanno come indirizzi gruppi di sedici bit.

Per poter gestire una memoria più grande, dato che in teoria non c'è relazione tra numero di bit di un dato e numero di bit di un indirizzo, si potrebbero avere dati di otto bit, ma indirizzi di trentadue e comunque più di sedici bit.

Come mai, quindi, non si sceglie questa soluzione per aumentare la capacità di memoria, e invece si ricorre a memoria di massa, come i nastri magnetici, i floppy disc o alle tecniche più sofisticate, tipo quelle delle memorie virtuali? Semplicemente perchè sarebbe, in un primo caso, più complesso, come vedremo in seguito, indirizzare oltre il valore 65536; in un secondo caso risulterebbe necessario cambiare l'architettura stessa della CPU, e di conseguenza, modificare la struttura dei linguaggi già dif-

Per informazioni e per collaborare:

• **Informazioni.** La Redazione è a completa disposizione per fornire notizie di qualsiasi genere ai propri lettori. Per ottenerle si prega di **telefonare** e non di scrivere, perchè il lavoro è tanto ed il tempo a nostra disposizione si riduce sempre di più. Purtroppo si ritiene comunemente che la telefonata "disturbi" e si preferisce, per discrezione, ricorrere a carta e penna. I lettori che scrivono però, pur se del tutto involontariamente, disturbano... moltissimo. Si pensi, infatti, al tempo impiegato dalla missiva per giungere in Redazione, alla fatica di aprirla e di interpretarne il contenuto, a rileggerla, perchè non sempre è ben chiara la richiesta, alla stesura della lettera di risposta o alla decisione di... cestinarla perchè di interesse non generale. Grazie al telefono, pur se con qualche lira in più, si ottengono informazioni in modo rapido e soprattutto si evita una gran mole di lavoro. I lettori interessati possono telefonare nei pomeriggi di giovedì e venerdì di ogni settimana allo 02/8467348.

• **Collaborazione con la rivista: invio di materiale.** Prima di spedire le vostre realizzazioni **telefonateci** in modo da stabilire, seppur sommariamente, se il vostro lavoro può essere interessante. Ricordatevi che (troppo) spesso molti lettori hanno fiducia in listati spaventosamente banali e trascurano invece di inviare realizzazioni di notevole interesse. Non siate modesti! Fate giudicare a noi se i programmi sono o meno da pubblicare!

In ogni caso ricordatevi che per facilitare il nostro lavoro:

• i listati sono accettati solo se inviati su supporto magnetico (nastro o disco). I listati su carta, non avendo il tempo di digitarli e quindi di controllarli, saranno **irrimediabilmente** cestinati;

• le prime cinque righe del programma inviato **devono** essere linee di REM di cui diamo un fac-simile:

```
10 REM *** PROGRAMMA "ASTRONAVI NELLO SPAZIO" ***
20 REM *** SOLO PER VIC 20 con 8K di RAM ***
30 REM *** DI LUIGI ROSSI ***
40 REM *** VIA GARIBALDI 43 ***
50 REM *** MILANO TEL. 02/1223344 ***
```

Il lettore, cioè, proprio come voi, vuol sapere subito, prima di leggere l'articolo, se il programma può girare sul proprio computer e, in caso di difficoltà, a chi rivolgersi per spiegazioni, oppure (tanto meglio per l'autore...) per acquistare il programma su supporto magnetico. Sono infatti numerosi gli autori di listati che hanno ricevuto richieste in tal senso da lettori un po' pigri che, scoraggiati magari dalla lunghezza dei programmi stessi, preferiscono acquistarli su supporto magnetico evitando la seccatura della trascrizione.

In conclusione... **il telefono, la nostra voce!**

fusi sul mercato internazionale.

Ritornando agli indirizzi ed ai dati, vediamo ora di individuare un indirizzo di cui sappiamo il valore solo in decimale o esadecimale.

Il problema si presenta quando si usano i comandi PEEK e POKE, dato che siamo costretti a fornire gli argomenti dei comandi stessi espressi come valori decimali, mentre spesso li conosciamo come valori esadecimali.

Conversione di un numero da un sistema ad un altro

Vogliamo convertire il numero 11052 decimale nel corrispondente esadecimale (che ha sedici simboli).

Si divide il numero in oggetto per il numero di simboli e si considera il resto ed il quoziente intero (primo resto = 12; primo quoziente = 690). Il quoziente, se maggiore o uguale a sedici, si divide nuovamente per sedici (secondo resto 2; secondo quoziente 43): poichè il nuovo quoziente è ancora maggiore di sedici, si ripete il procedimento finchè si ottiene un quoziente minore di sedici, (terzo resto = 11; terzo quoziente = 2).

Il numero esadecimale desiderato è formato dai tre resti ottenuti e dall'ultimo quoziente in ordine inverso: 2, 11, 2, 12.

Infine, sostituendo tali valori con i simboli corrispondenti in esadecimale (fig. 1 n.p.), si ottiene il valore cercato: 2B2C.

Convertiamo ora un numero esadecimale in decimale, trattando solamente numeri compresi tra 0000 e FFFF: la prima cifra rappresenta il numero moltiplicato per 16↑3: la seconda per 16↑2, poi 16↑1 e quindi 16↑0: pertanto, volendo convertire 2B 2C si scrive:

2x16↑3+Bx16↑2+2x16↑1+Cx16↑0

cambiando i numeri B e C esadecimali in decimali, otteniamo:

2x16↑3+11x16↑2+2x16↑1+12x16↑0 = 11052

PEEK e POKE

Prima di continuare è utile saper usare correttamente le istruzioni BASIC PEEK e POKE.

Queste istruzioni consentono di leggere (PEEK) in tutta la memoria e di scrivere (POKE): un qualsiasi numero intero compreso fra 0 e 255 in qualsiasi locazione della RAM. Per esempio, se noi battiamo:

PRINT PEEK (4080)

apparirà, in decimale, il valore della 4080ma locazione RAM. Viceversa battendo:

POKE 4080, 151

il valore 151 decimale sarà trascritto nella 4080ma locazione di memoria.

Da notare però, che, mentre il comando PEEK(X) legge un valore ed in nessun caso lo modifica, l'istruzione POKE X,Y cerca di modificare il valore della locazione X. Possono, infatti, verificarsi alcuni casi critici.

1/ Cerchiamo di scrivere in una locazione ROM del BASIC o del O.S. (Operative System; sistema operativo del computer), naturalmente, il dato che cerchiamo di scrivere non viene scritto, in quanto nelle ROM non si può scrivere e nessun messaggio di errore appare per informarci dell'impossibilità di eseguire l'istruzione.

2/ Analoga mancanza di messaggio di errore si verifica se l'indirizzo della POKE cade in una zona RAM non esistente nella configurazione, perchè il sistema, per esempio, da 8K, non è appunto espanso al massimo. In un PET da 8K l'ultima locazione utilizzabile è la 8191ma.

3/ L'indirizzo della POKE rappresenta una locazione RAM utilizzata dall'O.S. e una sua modifica può "distruggere" il sistema; saremo, in questo caso, costretti a spegnere e poi riaccendere il computer, perdendo, purtroppo, tutto il contenuto delle RAM.

4/ Il valore Y di POKE X, Y è negativo o è maggiore di 255. Questo è l'unico caso in cui appare un messaggio di errore. Teniamo presente, infine, che, se Y è un valore non intero, verrà presa in considerazione solamente la parte intera.

Supponiamo allora di non cadere in uno dei casi critici, come facciamo a memorizzare un numero maggiore di 255? Il procedimento è un po' lungo,

ma è l'unico che sia possibile adottare.

Vogliamo memorizzare il numero intero 11052 decimale. Abbiamo già visto che in esadecimale corrisponde a 2B2C, e poichè ogni cifra esadecimale rappresenta quattro bit, in totale avremo bisogno di sedici bit. Dato che ogni locazione di memoria contiene otto bit, noi potremo "spezzare" 2B2C in due gruppi: 2B e 2C; il primo, dopo averlo tradotto in decimale, lo scriveremo in una locazione, il secondo in quella successiva.

Riassumiamo il procedimento in altre parole:

```
11052 dec. = 2B2C esa.
2B      2C
2B esa. = 2x16↑1+Bx16↑0 = 2x16+11 = 43 dec.
2C esa. = 2x16↑1+Cx16↑0 = 2x16+12 = 44 dec.
```

Utilizziamo, per le verifiche che seguono, le locazioni RAM da 826 decimale (033A esadecimale), fino alla 1023ma, perchè tali locazioni sono destinate alla gestione della seconda cassetta e non rischiamo di distruggere il sistema. (Caso del Pet).

Scriviamo:

POKE 826,44: POKE 827,43

Come si può notare, del numero 2B2C, abbiamo trascritto nella prima locazione (826) il valore 2C (LSB: Least Significant Byte = Byte meno significativo) e nella successiva (827) il 2B (MSB: Most Significant Byte = Byte più significativo), anzichè al contrario, come saremmo stati indotti a fare istintivamente.

Adottiamo questo procedimento perchè, come abbiamo visto in figura 3 della prima puntata, lo segue anche la CPU. Come faremo, allora, a sapere quale numero è rappresentato da due locazioni di memoria successiva? Naturalmente seguendo il ragionamento inverso.

Traduciamo l'MSB in decimale e lo moltiplichiamo per 256:

2B esadecimale = 43 decimale; 43 x 256 = 11008

Analogamente ci comportiamo per l'LSB moltiplicandolo, però, per 1 (lasciandolo cioè invariato):

2C esadecimale = 44 decimale

In seguito eseguiamo la somma fra i due:

11008 + 44 = 11052

Altri esempi:

	LSB	MSB		
Loc. RAM	826	827	Significato	
	10	15	15x256+10	= 3850
	112	100	100x256+112	= 25712
	1	88	88x256+1	= 22529
	1	1	1x256+1	= 257
	255	0	0x256+255	= 255
	255	255	255x256+255	= 65535 (?)

Come si può notare il numero più grande che si può rappresentare è 65535 ed il più piccolo è zerò, ma tutti positivi. Se però rinunciamo ad ottenere valori così grandi possiamo seguire la convenzione secondo cui sono da considerare positivi i valori fino al numero (65535-1)/2 cioè da zero a 32767, mentre da 32768 a 65535 inclusi, sono negativi e valgono esattamente il valore considerato meno 32767:

Esempi:

```
52768 significherà -(52768-32767) = -20001
65535 significherà -(65535-32767) = -32768
32768 significherà -(32768-32767) = -     1
```

In questo modo possiamo rappresentare tutti i numeri interi negativi e positivi, compresi fra -32768 e +32767. Ecco spiegato, dunque, perchè certi computer non molto sofisticati (ed anche i Commodore quando usiamo variabili intere tipo A%) trattano solamente quei numeri interi: con due byte adiacenti non è possibile superare l'intervallo suddetto.

Guida mercato Commodore

Prodotto	Prezzo (IVA esclusa)

VIC - 20

Home Computer Vic 20 — 199.000
Unità di espansione *(1020)* — 295.000
Modulo di espansione *(1023)* — 135.000
Cartuccia da 3K di memoria *(1210)* — 66.000
Cartuccia da 8K di memoria *(1110)* — 98.000
Cartuccia da 16K di memoria *(1111)* — 172.000
Cartuccia Vic rel *(4011)* — 95.000
Permette di controllare il funzionamento di allarmi antifurto, porte automatiche, telefoni, trasmittenti ed apparecchi similari.
Vic switch *(4012)* — 225.000
Possono essere collegati fino a 16 VIC 20 con un floppy e una stampante (distanza massima 1500 mt.).
Interfaccia IEEE 488 *(T-1)* — 175.000
Interfaccia centronics *(T-3)* — 115.000
RS232-C adapter *(1011-A)* — 75.000
RS132-C adapter *(1011-B)* — 75.000

Commodore 64

CPU 64K RAM *(CBM64)* — 625.000
C 64 Executive *(SX 64)* — 2.350.000
Sistema operativo CP/M *(CP/M)* — 125.000
Consente di programmare il Commodore 64 in linguaggio CP/M, il più utilizzato sui Personal Computers. Permette inoltre di accedere alla enorme libreria di Software applicativi CP/M.
Pet speed *(6411)* — 95.000
Compilatore basic che aumenta la velocità di esecuzione dei programmi di circa 40 volte.

Accessori per Vic e Commodore 64

Stampante plotter a colori *(1520)* — 375.000
80 caratteri, per linea, 4 colori, alla risoluzione di 0,2 mm per passo.
Unità stampante *(MPS 801)* — 450.000
Stampa velocemente su carta normale quanto appare sul video: programmi, lettere, dati, grafici.
Unità stampante *(1526)* — 655.000
Stampante 80 colonne, bidirezionale, 60 CPS, spaziatura programmabile, trazione a frizione o a trattore.
Registratore dedicato *(1530)* — 120.000
Per memorizzare facilmente programmi e dati su normali cassette magnetiche.
Floppy disk drive *(1541)* — 615.000
Veloce unità di memoria di massa ad alta capacità. Può immagazzinare fino a 170.000 caratteri su ogni singolo disco.
Monitor monocromatico *(1601)* — 285.000
A fosfori verdi 12".

Per giocare

Comando per giochi (Joystick) *(1311)* — 13.500
Permette di muoversi in tutte le direzioni, di iniziare i vari giochi di movimento e di "sparare".
Comando a manopola per giochi (Paddle) *(1312)* — 22.500
Adatto per i giochi a 2 persone, esegue movimenti in orizzontale e verticale,

Commodore 4000

CPU 16K RAM *(CBM 4016)* — 1.285.000
18K ROM, BASIC 4.0 residente, video 40 colonne per 25 righe, tastiera semigrafica.
CPU 32K RAM *(CBM 4032)* — 1.495.000
18K ROM, BASIC 4.0 residente, video 40 colonne per 25 righe, tastiera semigrafica.

Commodore 9000

Doppia CPU 134K RAM *(CBM 9000)* — 2.350.000
Micro Main Frame Computer a doppia CPU (6502 - 6809) compatibile con tutte le periferiche Commodore della serie 8000. Include 5 linguaggi di programmazione. (COBOL, FORTRAN, TCL PASCAL, UCSD PASCAL, APL).

Commodore 8000

CPU 32K RAM *(8032 SK)* — 1.725.000
18K ROM, Basic 4.0 residente, video orientabile e basculante 80 colonne per 25 righe, tastiera commerciale separata.
CPU 96K RAM *(8096 SK)* — 2.285.000
18K ROM, Basic 4.0 residente, video orientabile e basculante 80 colonne per 25 righe, tastiera commerciale separata. Include sistema operativo PM/96.

Commodore 600

Indicato per applicazioni industriali, collegamento a strumentazione, controllo numerico, ecc. Utilizza monitor in commercio.
CPU 128K RAM *(610)* — 2.150.000
CPU 128K RAM espandibile internamente a 256K e esternamente a 960K, interfaccia RS232C, IEEE 488, porta utente a 8 Bit. Compatibile con tutte le periferiche Commodore della serie professionale.
CPU 256K RAM *(620)* — 2.550.000
CPU 256K RAM espandibili esternamente a 960K. Caratteristiche uguali al Mod. 610.
Monitor *(1601)* — 285.000
Monocromatico a fosfori verdi, 12".

Commodore 700

CPU 128K RAM *(710)* — 2.850.000
CPU 128K RAM espandibili internamente a 256K ed esternamente a 960K. Video orientabile e basculante 80 colonne per 25 righe. Compatibile con tutte le periferiche Commodore delle serie professionali.

Riservato
agli ingegneri

Il miglior software tecnico su elaboratori CBM - Commodore Ora anche disponibile su Commodore 64

"S.S. - 80"

L'ormai famoso programma per il calcolo delle strutture intelaiate piane in c.a., in zona sismica, che sviluppa e disegna anche le carpenterie delle armature.
(Ultima versione Luglio/1982 nostra esclusiva).

"FONDAZIONI"

Risolve tutti i problemi di fondazioni (trave elastica su suolo elastico) di strutture in c.a. in zona sismica e non, risolvendo l'intero graticcio di fondazione e proponendo una carpenteria sofisticata ed ottimizzata.

"MURI DI SOSTEGNO"

A gravità, a mensola o a contrafforti, anche in zona sismica, secondo il D.M. del 21/1/1981.

"PENDII"

Analizza la stabilità di un pendio o di un fronte di scavo sotto diverse condizioni e la verifica relativa viene condotta in termini di tensioni effettive; la stima dei fattori di sicurezza viene effettuata secondo i metodi di Fellenius, Bishop e Jambu.

"COMPUTI METRICI"

Analisi ed elenco prezzi Metodo veloce e completamente automatizzato per il computo e la stima dei lavori.

"REVISIONE PREZZI"

Secondo le disposizioni di legge vigenti. Praticità ed automazione consentono di eseguire velocemente revisioni di prezzi anche per lunghi periodi.

Richiedeteci documentazione e output dei programmi di vostro interesse. Resterete sbalorditi dalla versatilità e dalla completezza del nostro software.

SIRANGELO COMPUTER Srl
Via Parisio, 25 - Cosenza 0984-75741

NEW NEW NEW NEW NEW NEW NEW

È pronto il nuovissimo programma

"ORARIO SCOLASTICO"

CPU 256K RAM *(720)* 3.250.000
Monitor a colori 14" con audio. *(1702)* 645.000

Dischi

Floppy disk drive *(2031)* 675.000
Unità di memoria di massa ad alta velocità. Capacità 170KB. Drive singolo.

Floppy disk drive *(4040)* 2.095.000
Unità di memoria di massa ad alta velocità. Capacità 343KB. Drive doppio.

Floppy disk drive *(8050)* 2.375.000
Drive doppio 1M byte in linea.

Floppy disk drive *(8250 H.P.)* 2.450.000
Drive doppio, doppia faccia, doppia densità 2M byte in linea.

Floppy disk drive *(8250 L.P.)* 2.600.000
Drive doppio, doppia faccia, doppia densità, 2M byte in linea.

Hard disk *(9060)* 6.200.000
Tecnologia Winchester, .5M byte in linea.

Hard disk *(9090)* 6.700.000
Tecnologia Winchester, 7.5M byte in linea.

Stampanti

Stampante *(4023 P)* 695.000
Bidirezionale ad aghi, 60 CPS, 80 colonne.

Stampante *(MPP 1361)* 1.275.000
Stampante ad aghi 150 CPS, 132 colonne, bidirezionale, trascinamento a trattore.

Stampante *(6400)* 3.250.000
Stampante a margherita, 40 CPS, 136 colonne passo pica, 163 colonne passo élite, bidirezionale, utilizzabile anche con carta da bollo, trascinamento a frizione o a trattore.

Accessori

Microprocessore 32K RAM *(MUPET II)* 2.500.000
Per connettere, in rete fino a 16 CPU RS232, IEEE 488, centronics. Il prezzo include (configurazione minima): controller, terminator, 3 moduli, cavi, cavo IEEE/PET (per la versione SK).

Singolo modulo addizionale: – 325.000
1 modulo
1 cavo 6 piedi.

Nuovo sistema operativo *(PM 96)* 95.000
Per 8096SK o per 8032SK con B - 1 oppure con B - 2. Può gestire fino a 16 programmi residenti simultaneamente in memoria. Da a disposizione 26K per le variabili e 53K per i programmi. Potenzia inoltre il Basic con altri comandi.

64K RAM *(B-1)* 575.000
Scheda di ampliamento memoria per 8032 e nuovo sistema operativo "PM 96".

CP/Maker *(B-2)* 1.450.000
Incrementa la memoria interna di 64K RAM e permette l'uso di tutti i programmi CP/M. 8 bit disponibili. Compatibile con la serie 3000/4000/8000.

Scheda ad alta risoluzione grafica *(B-3)* 720.000
Compatibile ai sistemi della serie 8000.

Cavo PET/IEEE 488 *(C-1)* 85.000

Cavo IEEE 488/IEEE *(C-2)* 95.000

Accoppiatore acustico *(8010)* .595.000
300 baud/sec.